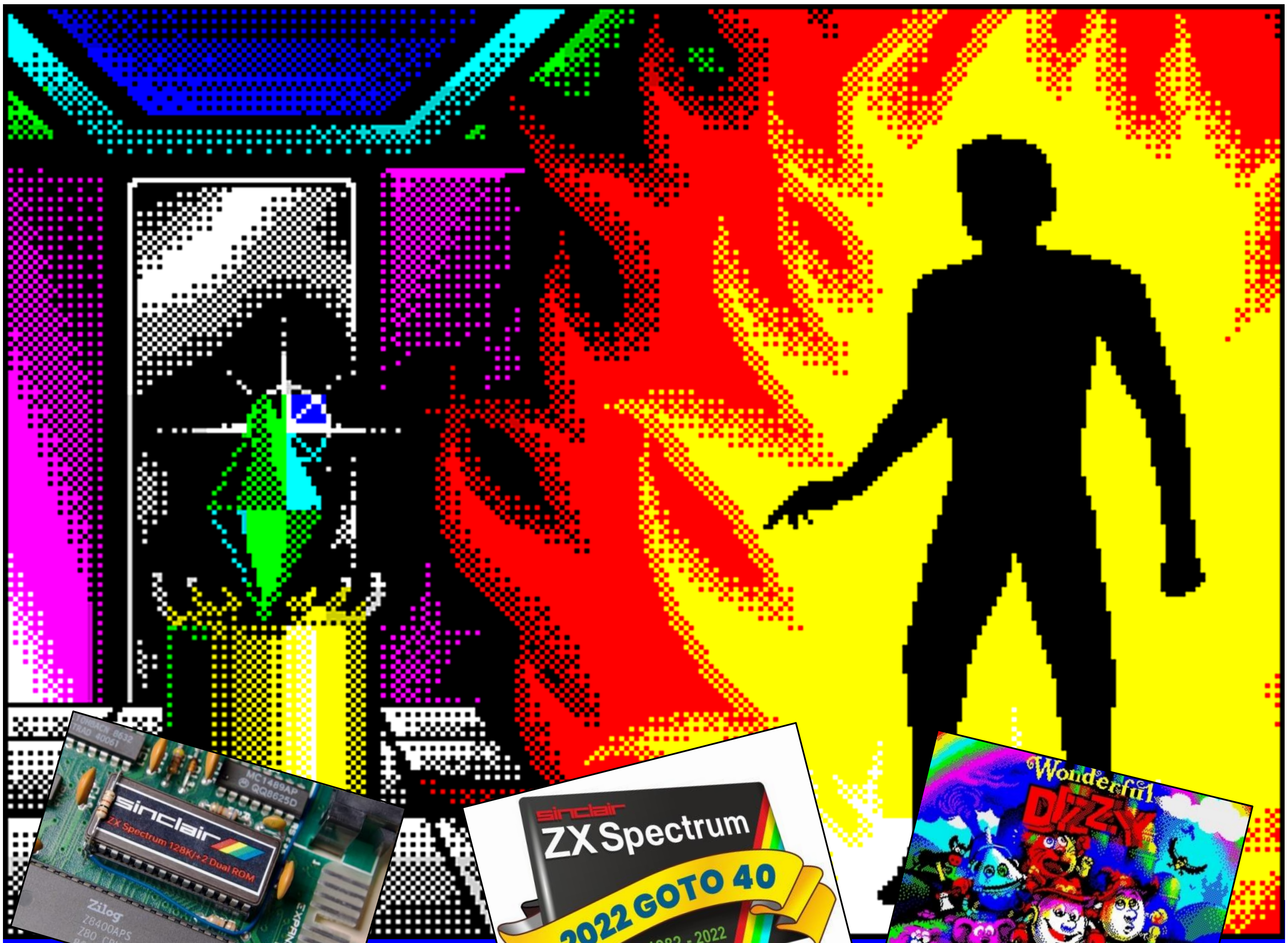
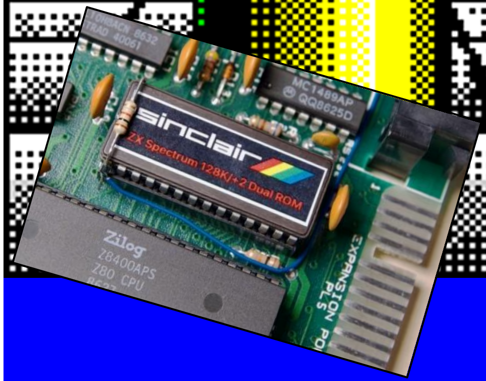
SUC-SESSION

YOUR MAGAZINE FOR THE SINCLAIR SPECTRUM





- News
- Dual-ROM for the Spectrum +2
- Game hack »*Monjas*«
- Adventure solution »*Calling*«
- ZX SPECTRUM48 System variables
- Computer Classics *Sharp MZ700 & MZ800*
- And much more …

ISSUE
231.
OF YOUR
MAGAZINE

FOR SUBSCRIBERS WITH SCENE+ DISK

THE NEWEST AND BEST PROGRAMS
DIRECTLY DELIVERED TO YOU

EDITORIAL

Hello dear SUC readers

it has been almost four months since the last SUC session was published. Many things have happened since then. Of course, we are all preoccupied with Russia's unfortunate conflict with Ukraine - the news is full of it. It seems that an autocrat wants to live out his dreams of great power. What annoys me personally is that once again an autocrat has managed to draw an Iron Curtain across Europe - only this time further east. When the Iron Curtain fell 33 years ago, everyone believed that the permanent conflict between East and West was over. But we were obviously mistaken. The lessons of World War II have apparently been forgotten. Helmut Schmidt, our former Chancellor, said at one point, »Better to negotiate 100 hours without success than to fire a single shot.« I can only agree with that and hope that it will all be over soon.

Fortunately, at least the Corona pandemic seems to be coming to an end. The current virus variant, while highly contagious, apparently causes a less dangerous course; at least according to the media. But that is no consolation to those who have lost loved ones or are suffering from Long Covid.

Enough of the nagging!

Let's look forward to the current issue, in which **Frank Dopierala** explains how to install a double eprom in a Spectrum +2, **Thomas** reveals the first secrets of his hacker career and **Harald Lack** introduces us to the solution of the game **»Calling«**

Plus, ***Spectrum40*** is just around the corner. **Thomas Eberle** and **Lee Fogarty** invite you to the ***Spectrum40 Event*** on **Saturday, April 30th** in Walsall at the *Bescot (Bank's) Stadium*. On the map, which can be found on the website, you can also see the American Embassy - and about four miles airline distance there is a McDonald's...

All information can be found on the English website. The link to the website is:

`https://spectrumevents.website/`

Finally a call for help!

We, the editorial staff of the **SUC-Session**, are still looking for someone, male, female or mixed, who feels called to take over the game reviews of the **Spectrum Next**. Please appear numerous - Thomas will also pay you for a Creamed Tea at the next meeting!

Jokes aside! We have reached the copy deadline, so for this reason there is no article about NEXT-Basic in this issue. We can't write everything ourselves. There are other things in life than the **ZX Spectrum** - even if this is hard to believe.

Anyway...

We wish you all a lot of fun with the magazine.

Many greetings,

Joachim and Thomas

April 11nd, 2022

CONTENT

IMPRINT

The **SUC-Session** magazine appears in German and English and can be subscribed together with the disk-magazine **SCENE+** at the **»Spectrum-User-Club«**. For further information please contact Thomas Eberle.

Editor SUC-Session 227:
Joachim Geupel & Thomas Eberle

Editor SCENE+: Thomas Eberle

Club manager:
Mirko Seidel, Adresse: Birkenweg 8
89197 Weidenstetten
email: mirko.seidel1@gmail.com

This issue has been supported by:

Joachim Geupel	Joa.Geu@gmx.de
Thomas Eberle	thomas.eberle@sintech-shop.de
Ellvis	
Harald Lack	
Frank Dopierala	
Dieter Hucke	http://www.dieter-hucke.de/

<u>**Copy deadline for SUC SESSION 232**</u>

Wednesday, 29 June 2022

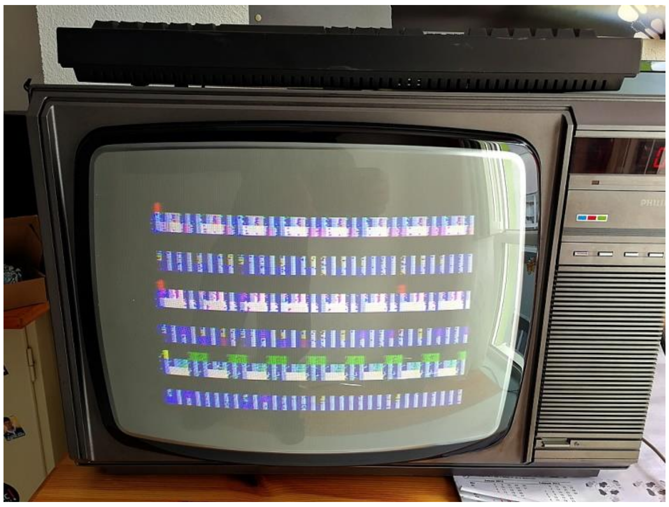
READERS LETTERS

Reinhard Hock from Trossingen/Germany has unfortunately a defect Spectrum +2A.

Unfortunately I had recently a problem with my Spectrum +2 (A) 128k, maybe you have some advice for me: I plugged in a Divide module and then got a strange screen, see enclosed picture.

In the meantime I removed the Divide Interface again. When I switch on the Spectrum, however, nothing happens. What could be the reason or how could this be fixed?

Many greetings, Reinhard Hock



Hello Mr. Hock,

I am not sure, but many interfaces, also some DIVIDE are not made for the Spectrum +2A, because it has a different peripherial port pinout than the predecessor Spectrums. This differently pinout can lead to a defect of the computer. It is of course also possible that the interface is basically suitable for the +2A, but e.g. something was plugged in crooked or was plugged in the Spectrum during operation. All these things can lead to a short circuit and because the Z80 is directly connected to the port, very often the processor is broken, in few but bad cases also the ULA can be affected. A defect ULA I would describe as a total loss with the +2A, the Z80 on the other hand is easy to replace. At SINTECH you can also order a Z80. The Z80 should be socketed and is easy to replace, I think it is worth a try at this point.

Reinhard Hock from Trossingen has contacted me again with questions about the hardware.

Hello Mr. Eberle,

attached is a photo of my Z80. Can you tell from it if the chip is socketed? I'm looking around for a replacement Spectrum and am considering between a 48K model as well as another +2 128K. However, I don't really trust my Divide Interface anymore and am therefore looking around for a possible alternative. In your online store I noticed the DivMMC interface. Is this similar to the Divide or even a bit more comfortable? I have the following questions:

- *Is it possible to connect both the VGA/joystick adapter and a DivMMC module to the Spectrum using the ZX-EXT I ordered and thus use both together?*
- *Will 128K games run on a 48K Spectrum using the DivMMC? (Probably not...)*

Thanks in advance for your answers,
Reinhard Hock

Hello again, Mr. Hock.

The Z80 chip on the picture is socketed, so it is not soldered in. It can be carefully lifted out and then a new Z80 inserted. You have to make sure that all feet are in the socket. Pay attention to exactly the same direction.

No matter if Divide or DivMMC, both work together with ZX-EXT and ZX-VGA. The DivMMC has the advantage that it is also useable for the +2A; with your Divide I was worried, if that could also be the reason of the crash.

To play 128k games you also need 128k memory, a Divide or DivMMC does not expand the working memory of the Spectrum. Also a 48k Spectrum has no AY sound chip. For this we have also another interface, but at some point there are enough interfaces.

A third time in this issue Reinhard Hock from Trossingen, this time with a keyboard problem:

Hello Mr. Eberle,

I bought a Spectrum 48 k+ the other day, the one with the »real« keyboard, a really nice computer. ☺) Interestingly, the keyboard only works completely when I unscrew the unit and lift the top part with the keyboard a bit from the bottom part of the case. If I then put the top back in the correct position, a few keys no longer work. I assume that this is due to the transparent membrane that connects the keyboard to the motherboard? If so, it shouldn't be that difficult to replace it? Fortunately, my Divide module works fine with the Spectrum +. ☺

Many greetings,
Reinhard Hock

Hello Mr. Hock,

yes, the membrane probably has a fine crack which closes in a certain position of the membrane and breaks the connection in normal position. Changing the membrane should solve the problem.

The installation of a membrane is not very easy, the foil consists of several layers which have to be pressed together. Therefore the screws have to be tightened in exactly the right strength, finding this out is a bit of a search game. But with a little trial and error, you can do it.

The fact that the Divide now works proves that something is wrong with the other Spectrum.

Paul Veltjens from Geilenkirchen/Germany contacted me by e-mail about the planned discontinuation of the SCENE+:

I don't have any disk drives and so I don't need any. However, for sentimental reasons alone, I *would find it a pity if there were no more disk version. My suggestion: reduce the number and either make it more expensive or give it away to the highest bidder. This might make it more interesting. Kind regards Paul*

Hello Paul,

it is not about copying disks, but about creating the DiskMagzine. All cassette loading commands have to be changed to floppydrive commands. Sometimes you even have to rewrite MC loader routines or even move the whole code because of too high RAMTOP. In addition the whole code is still memory-technically optimized, by packing programs the allocated RAM Space is reduced to a third shorter and so we can put more programs on a disk. You don't have such worries with SD cards, they work with tape loading routines and have no memory problem (at least none where 15 kB is important). For the Scene+ I spend a lot of time and energy, which I just want to use in the best possible way. I'm sorry too, but I dare say that the vast majority of people don't subscribe to our magazine because of the disk. It just doesn't make sense to put so much energy into a product that is hardly in demand. I'm happy to be proven wrong, but that's how the situation looks to me at the moment.

Lothar Ebelshäuser from Kall-Scheven/Germany would regret the discontinuation of the SCENE+ very much:

Hello Thomas, again something from me.

Of course I would like to be a member of the club further. The magazine has arrived, and again has become really good. All articles are well done and easy to understand. Now I am looking forward to the next disk. And that is already my main concern:

You already mentioned this topic in the last issue. In so many years, of course, a lot of disks have accumulated from the Scene+. Again and again I take out one or the other disk and play for some time.

It would be very, very difficult for me to do without the mag. If the Scene+ would be also available on SD card for the Spectrum Next, it would again be a reason to get more involved with the NEXT computer.

But both would probably be too expensive economically, apart from all the work. So please continue with the disks, they are getting better and better. And I'm always looking forward to the next one.

I don't know much about TAP files.

Many greetings from the Eifel, Lothar

Hello Lothar, thanks for your message. It is of course nice to read that the created SCENE+ is also in use, but honestly I have the feeling that many people have the SCENE+ just because it comes with the magazine. But first and foremost, most people are interested in the magazine, and hardly anyone uses floppy disks anymore. Of course, we could also publish the SCENE+ on an SD card, but I don't see the point. On the SD card there would only be a TAP file. Everybody who has an interface for SD card or a Spectrum NEXT etc. also has an SD card and usually they have 4GB. On it fit not only all SCENE+ outputs ever created, but generally the entire Spectrum software library. So instead of using a lot of SD cards, a user could just copy the TAP file to his already existing SD card, he could create subfolders and have it nicely sorted without having to change the SD card. With this in mind, it then becomes apparent that the SCENE+ as TAP file is then compelling unnecessary. The disk versions are packed to save memory space and adapted to the load syntax. Both is unnecessary with TAP, there is plenty of RAM space as already mentioned, and the syntax of cassette remains the same.

My idea is instead, a club website accessible to all with the program links of the programs we recommend. So everybody can create easily his own SCENE+, any user can then focus only on games or only on demos. The question is, are there still users who have a +D/Opus and no interface for SD cards? I doubt it. If there should still be someone, they would fall by the wayside a bit, although you can load the TAP files via emulator and copy them to floppy disk, but you would probably need at least a WIN2000 computer with a floppy disk controller. And who else has such hardware nowadays...except me? So the question is: How many users really NEED the SCENE+, because they have no other possibility to transfer software. And then the question, if it's only one or two users, can't they switch to a DivMMC after all as a more modern, safer and ultimately easier storage? Gladly I read to it.

Norbert Opitz from Wittenberg/Germany also has a suggestion for the continuation of the SCENE+:

Hello Thomas !

Make the disk magazine Scene+ as a TAP file and the SUC session as a PDF file, so that you have less work for these information media.

Bye, Norbert

Hello Norbert,

thanks for your idea. It is important to know that neither Joachim, nor Mirko or I are doing all this for money, the contributions are primarily to cover the costs. PDF and TAP also cost a contribution, but these contributions also serve to cover costs, even though the costs may be incurred elsewhere. Distributing the magazines digitally only would cut almost all costs and we could distribute the magazine for free. However, I wonder if a free magazine would still be of such value to readers. The Internet is full of free information, but who uses it? We think that there are users who want to have a printed magazine in their hands and pay more for it. But everyone is free to collaborate and get the magazine cheaper or even for free. This is our concept to make sure that cooperation is rewarded and that we really

get help. Joachim and I cannot fill the magazine alone, we would then only have an 8 page information sheet. As for the SCENE+, all the programs we publish are already available as TAP, we provide the links in our game and demo reviews. A pure TAP edition seems needless to me, the TAP was a side product when creating the MB02 version, but if there is no more MB02 - version, there will be no more TAP.

Harald Lack from Raubling/Germany contacted us in another matter, which was very embarrassing for us, but which we do not want to hide:

Hello Thomas,

I just heard from Joachim that I am missing the last three issues. Didn't you always send an email when the new issue was out? Do you have my correct e-mail address? Joachim sends me the three issues. Just for your information!

Greetings Harald

Hello Harald,

indeed I have forgotten our most diligent reader. Actually it is like this: You get a subscription by paying. For your articles you get points, which you can use for payment. But the subscription does not renew automatically, you have to proactively renew the subscription and use your points to pay.

This is how it is generally intended. In your case I probably didn't send you a reminder, you are in every issue anyway, so there is no point in making a list. But in theory there is this list and so it went wrong. This is just to explain how it could happen. Joachim has sent you the missing booklets, apologies from my side.

Josef Prokas from Sibirna/Czech Republic renewed his subscription:

I would like to renew my subscription :-)

I sent you some contribution payment for all the PDF you sent me from previous times. Thanks a lot. The TAP is perfect for me, because I do not have any older, legacy device.

I am a pure esxDOS user (with MB03+ and eLeMeNt ZX), concentrating mainly on my zxfiles.samcoupe.cz website.

(I also updated sam.speccy.cz and now have to update esxdos.samcoupe.cz)

I believe, after years we should join the most popular speccy interfaces and add-ons to one machine.

LMN128 has added Sega Controller and Pentagon-4096 to its eLeMeNt/MB hardware, this is the Link:

https://oldcomp-cz.translate.goog/viewtopic.php?f=129&t=8898&start=45&_x_tr_sl=cs&_x_tr_tl=de&_x_tr_hl=cs&_x_tr_pto=wapp#p136560

I am still working on new versions of manuals for the eLeMeNt ZX, LnxCopy, LnxCommander... and technical docs.

As far as I communicate with my russian friends, more people from Russia are interested in the western types of speccy add-ons. LMN128 already implemented TurboSound, GeneralSound, SoundDrive, Pentagon 1024, Z-Controller.

And now the russian spectrumists produce Pentagons with DivMMC!!! Especially the Sizif-512 looks great.

https://sam.speccy.cz/zxtoday.html

The esxDOS with its TRDOS and tape emulation layers is perfect for unifying various software for different systems.

Second part of my activities is conversion of old apps for the esxDOS. Eg. Datalog or Text Machine. Yes, I am still using this program at work :-) https://sam.speccy.cz/datalog.html

I am not sure if there is some bigger demand for these activities. If so, please let me know, I could write an article to your mag. However, according to my experience, people are stucked in the past and it takes them many months or years to adapt for actual, better add-ons.

And SpecNext, of course, it brought - imho - people to another, amiga-like platform...

Greetings, Josef

Thanks Josef for the insight into your extensive activities. Indeed there was once a small but fine community for the SAM Coupe also in Germany, I also think that some of our readers still have a SAM. But I didn't really feel the SAM as a Spectrum successor, because not even the bank switching works the same as with the Spectrum and therefore Spectrum 128k programs are not compatible. Only the 48k mode can be emulated. The Spectrum NEXT has a slightly different approach, at least all Spectrum programs work for once and the Spectrum NEXT can also be used like a Spectrum. It is also a worthy successor in terms of design. That the NEXT offers even more, which is partly similar to the Amiga, is right, but in my opinion only a few people use it. In fact, one thing has remained the same: The NEXT is, similar to the SAM, easy to program in Basic and this distinguishes both SAM Coupe and NEXT very significantly from the Amiga. In this respect, both devices can be seen as successors of the Spectrum from the point of view that the operation is quite easy to learn. The Spectrum NEXT just in a somewhat newer version and with its own standard.

In fact, there is not known a lot in Western Europe and the rest of the world about what is being developed in the Czech Republic and Russia. Your website gives some insight, but I think I speak for everyone when I say that we would like to hear more about it, especially the core question: where can I get this great stuff? So we love to hear more from you.

Letters to the editor as always to
thomas.eberle@sintech-shop.de

NEWS IN SHORT

NEW SOFTWARE AT SINTECH

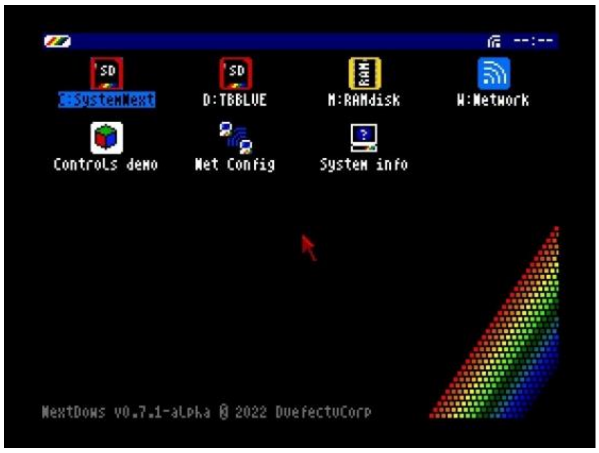
Sintech in Germany now has a section for new Spectrum games. For this purpose **Sintech** has taken over the distribution of the games of **Sebastian Braunert**, currently these are »Moritz the Striker« and »Moritz on the Autobahn«. **Sintech** is also working on the distribution of further titles.

There are also plans to expand distribution at **Sintech.UK** and **Sintech.CZ**. The titles in the German store can be found here :

`https://www.sintech-shop.de/retro-atari-commodore-sinclair-etc/sinclair/spectrum/neue-software`

Due to the special Brexit rules **Sintech-Deutschland** does not ship to UK and due to the war situation also not to Ukraine or Russia.

SPECTRUM NEXT OS "NEXTDOWS



An interesting operating system for the **Next** is currently under development. (actually v.0.3 from March 22nd). After booting you will find yourself in a **Windows** -like (or let's say **Mac OS**, that sounds better) interface with icons and folders, as we know it from said system. Included is also a File Explorer for copying, deleting and executing files. Some people are suspicious of such an interface, but I can reassure all of you who think it is not *»retro«* enough, because the beta disk interface already had such an interface (vision) in 1984. In comparison to that **»NextDows«** is much more extensive, offers also e.g. network settings, sound settings, screen settings, time... The whole thing is still in beta phase with version 0.7.1 . Only those who support the project financially can learn more and test it (the amount can be chosen freely, even 1 Euro is possible). You can offer your support on the following page:

`http://zx.duefectucorp.com/2022/02/19/support-us/`

We will surely inform about this project in more detail when a complete version is released.

FURTHER DELAY FOR SPECTRUM NEXT KS2

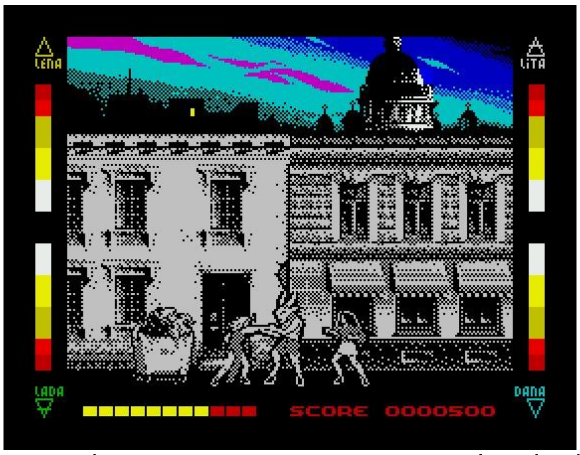
The worldwide chip shortage does not pass us retro-users by. For the **Spectrum Next** this means a further delay, because the programmable FPGA chips »**Spartan-6**« are no longer available on the market. The team had to redesign the whole board to run on the more powerful (but also more expensive) **Artix-7** chip, which is apparently better available on the market. So currently, the team now has two designs ready to go and is hoping to get the right chip in sufficient quantity for one design.

Unfortunately, this will require different firmware for the **Spectrum Next** from this production, this can't be helped.

FILMMAKER ANDY REMIC DECEASED

The author of films like "MEMOIRES OF A SPECTRUM ADDICT", Andy Remic), had also planned a film about the development of the SPECTRUM NEXT. It will probably not come to that, at least not from Andy. He passed away on February 26th 2022 for reasons unknown to us at the age of only 51. Rest in piece, Andy.

ZOSYA PUBLISHES SCREENSHOTS



Due to the situation in Russia, **Zosya** has had to stop distributing **Spectrum Software** for the time being. This has not stopped the team to continue programming and the game »**Rubinho Cucacaracha**« will be released online soon. Until then, the team passes the time and publishes screenshots of about 100 other projects, which promise a lot of great graphics and action. The best way to find **Natasha Zotova's** posts is on Facebook:

`https://www.facebook.com/groups/164156683632183/search/?q=natasha%20zotova`

SCENE+ Disk No. 76 German

By Thomas Eberle

Those who subscribed in time had already received Diskette No. 76 (or TAP) before this issue. Again there were many great programs on it:

- Oxygen - scene demo
- Karateka - scene demo
- 70908 - scene demo
- Valley of Rains - action game
- Cell3326- maze game
- Block Dude - puzzle game
- The house on the other side of the storm - text adventure game
- Just a Gal - racing game
- Yoyo´s great adventure - action adventure game
- Void - scene demo
- Space Invaders - arcade baller game
- Coloristic - puzzle game
- Mire Mare - maze game
- Monjas - action adventure game
- The Humans - puzzle game
- Agent Blue - platform / action game

At first we want to thank all the authors, especially ZOSYA Entertainment for the permission to use also the actually commercially distributed games. All in all a colorful selection of mainly games of all genres, mixed with the best demos. Unfortunately, this time there are no sound or graphic shows and as so often, no utilities (they are rare). Nevertheless a successful issue, whose realization was also not so easy.

This is unfortunately also the reason for the following lines: The conversion of games to floppy disk is difficult and time-consuming. It's never done simply by changing the Headers BASIC. Often the loaders of PD and commercial programs are protected, in almost all cases the programs are not memory optimized and have to be packed to fit more on a floppy disk. But the biggest problem is of course the conversion to +D and then also to Opus. Both floppy interfaces occupy some RAM in the main memory of the Spectrum and very few programmers take this into consideration. Optimization is only done for cassette or maybe for TR-DOS (Russia), but not for Opus and +D. I don't mind the work, but the less amount of users. For Opus we still have 4 subscribers, few more for +D. All others choose the TAP format. On TAP format you don't have to change much on the programs, it's very easy to do. The suggestion was that we simply make only the TAP format. But I don't think this makes sense, the programs are all available on the internet in TAP format, why should I do the work to summarize them, when Ellvis describes games and demos in every SUC session including a download link.

Since the subscriptions are still running, there will be another issue of SCENE+. With issue 77 we will stop this magazine and concentrate on the SUC-Session.

Nevertheless, I would like to open the discussion about alternatives. How about a website to download disk conversions? Who is interested? I would like to hear from you.

Feel free to write us a letter or email.

From the Article to the Magazine

A Personal Review

By Joachim Geupel

The SUC Session has been created on my computer for a little more than three years. I have not counted the number of articles I have contributed, but there are quite a few. Generally there are at least two, usually there are more.

I got to this by accident and because I'm a good person - at least most of the time. And it happened like this:

On the way back from the first Spectrumania I attended, Mirko and Thomas tried to convince me to take over the SUC session, which I categorically refused. I carelessly told them that I had edited the club magazine for two clubs or societies several times before. The first time was between 1985 and 1989, when private mailboxes and remote data transmission by means of acoustic couplers and sometime a highly illegal Hayes modem were the order of the day. As real German club-men, we, the users of a mailbox called Zeropage, founded a club that wanted to popularize remote data transmission. Which worked out. The second and the third time I was connected to a SF club, which dealt with a series of the 60s, a little more precisely, namely Ren Dhark[1]. Here I even did the duplication and shipping at the end. The effort was immense and deeply interfered with my private life.

[1] The copyright of Ren Dhark is reserved by HJB publishing house

At some point it was no longer fun. I had a lot of work and was only criticized. When the point was reached where the annoyance outweighed the fun, I stopped my involvement. In the end, I even buried the club. Fortunately that will not be the case here in the Spectrum User Club so far and it is still fun.

Thomas eventually managed to convince me to take over the magazine. I resisted for a long time! Honestly! But now I'm driving a Ferrari financed by Thomas! Really!

Of course not! Thomas has so much work to do that he hasn't been able to keep up with the magazine. And because Thomas has become a good and trustworthy friend in the meantime, I wanted to help him. Ok, however, a baseball bat is a convincing argument....

My objective was and is to publish the magazine regularly and to make it readable. I think I have succeeded - at least no one has complained so far. Thomas and I have an excellent working relationship. He sees the mistakes that I don't see because I'm much too close to the magazine. You don't see your own mistakes until the issue is printed... If I'm too direct or too nasty in the foreword or in an article, he defuses them. I often envy him his calmness.

Thomas. Consider yourself praised!

WRITING ARTICLES

I've never actually thought about writing about the grammar and spelling of the German language. However, I can't seem to get around it.

I have two requests for you!

Of course! Write articles! Otherwise, the magazine will be too thin. Our goal is to make between 36 and 44 pages. To do that, we need contributions. And admittedly, I'm running out of material. I soon can't think of anything new, at least nothing Spectrum-specific. That I present the computers of my collection, because the time between 1980 and 1986 was very interesting, you have surely noticed yourselves. I do research on the respective computers, contribute - if available - personal experiences and present them to you. But eventually I will have presented all my interesting devices and this series of articles will end.

So: write articles!

The second concern is actually the articles I get.

Most of the articles are fine. It's a pleasure to read them and I'm glad to see that there are still people who also deal with a computer whose highlight time was actually the early 80s. I sit here in the most exposed position because I get to be the first to read the articles that come from you. This is exactly where the work begins for me.

It takes me between three and seven days to produce an issue. Depending on what else I have to do, I spend between three and seven hours at the computer. By the way, I create the booklet with Word 2010, which is actually very bad for this. Word is okay, but it's not a DTP program. But I know it so well that in the end it will produce the booklet you are holding in your hands.

My problem is that I need more time to revise the articles, because there are sometimes many spelling mistakes in them, or I get texts whose formatting I can't take over directly to create the booklet. I usually see the spelling mistakes at first sight; I correct them - done! In fact, texts with the idiosyncratic formatting are the real problem for me. Below I explain how I would like the articles to look.

Let's start from the beginning.

WHAT I WOULD LIKE TO HAVE:

VERY IMPORTANT! HEADLINES!

Every article needs a headline. The headline gives a first clue what the article is about. If it is missing, I - or the other readers - cannot estimate what the article is about. It may be a game description or the introduction of new hardware, or it may be a report about a meeting or the personal experiences of a reader who introduces himself. I can't know that beforehand. Accordingly, I read through the article and give it its own headline, which, if I'm unlucky, misses the point and then incurs the author's displeasure. That already happened, though not in the Spectrum User Club.

So: a headline about the article, please.

ALSO STILL IMPORTANT: INDICATE AUTHOR OF THE ARTICLE.

I keep getting articles that are highly interesting, but I have no idea who created them. I am happy about every article, but I should know the author. The first name or only the initials of the name are not enough.

Of course I take into consideration if the respective author does not want to have his name mentioned; we will respect this of course, but however, I should know at least internally the authors name, so that I can inquire in the given case if there are ambiguities or questions.

PARAGRAPHS!

An article should be divided into paragraphs. By this I do not mean different paragraphs that introduce a new topic in the article. I mean that: when an idea covered in the paragraph and covering a topic is finished, but the topic is not finished, a paragraph must be inserted.

<u>Example</u> from issue 230, Clive Sinclair, section »The Years After«:

> *(...) In this poker round, he was eliminated first, but won the first season of Celebrity Poker Club in 2003 and took home £25,000.*
>
> *Sir Clive Sinclair received several honors for his contribution to the establishment of the personal computer industry in England. (...)*

Here, the years following *Sinclair Research Ltd.* are covered, moving from one stage of life to another. So, in writing, one train of thought was finished and another was taken up. Paragraphs make reading easier and the text more understandable.

ANOTHER DIFFICULT SUBJECT: NUMBERING AND BULLET POINTS!

The most vary enumerations of the different articles make my life really difficult.

Example:

*** .... Text**

(here is some explanatory text. But because I don't want to expose anyone, here come two lines of gibberish from me, followed by the letters once I lean across the keyboard: +äüpolörgnmfklawäplsfgml .ökjhtgvöcsoaöml)

*** ...... Text**

(These lines are just filler text with no meaning. So take cover! You too, Thomas!)

(...)

I have to remove all these bullets by hand to replace them with the appropriate function from Word. Likewise, I then have to replace the indentations.

Word offers the possibility to automate enumerations. To keep the layout reasonably even, I use these formatting options, be they numbering or just bullets.

What I would like is that, should numbering be necessary, there is only a number at the beginning of the line, e.g. »*1.*« or »*1.)*«, without inserting any text characters like *punctuation, +, **, etc.. Here a tabulator offers itself. It is inserted with the Tab key (⇨| →|) and has a fixed spacing. All text programs known to me, even the Windows editor, can set tab stops. Instruction characters, asterisks and other puntuations are unnecessary.

THE LENGTH OF THE ARTICLE

In recent issues, there have been articles that required three issues of *SUC Session* to even appear complete. These articles need to be split up. I find it tremendously difficult to find the sections in such an article where I can split them up. If such an article is 20 pages long, it has to be divided into at least three, better four parts.

In principle, I have no problems with long articles, however, I am not willing to reserve so much space in an issue to publish all 20 pages at once. If I have to make the division, there is a risk that the text will lose consistency and the readability will suffer.

WORDING AND STYLE OF THE TEXT

There is little to say here. In general, I leave the texts as they are. There are exceptions, of course, namely when I read the text and have to assume that nobody but me wants to finish it. In that case, I send the text back to the author and ask him to reformulate the text.

Write the way you want to read it. When the text is finished, leave it for two weeks and then read it again. It is worth it! Spelling errors will be caught, wording will be revised, commas will be put in where they are not, and so on... Give it a try.

IMAGES:

I have problems processing RAW images, i.e. images in the respective raw format of the corresponding digital camera. I have to convert them extensively and sometimes reach my limits. In the meantime, I already have some tools that allow me to convert them into a format that is understandable for Word.

The pictures should arrive best in JPG format, compression factor 100% (no compression). For your information: the smaller the percentage value is, the more compressed the image and the quality is. The image size is not important. Nevertheless, I prefer the size 3888 pixels in width and 2592 in height with 300 dpi resolution.

If you want to include pictures in the article, please limit them to a maximum of five.

EDITORIAL DEADLINE

The editorial deadline is usually the time when I start working on the issue. The later I receive the texts, the later the issue will arrive and the more hectic it will be for me. Please (!) keep to the editorial deadline, otherwise your contributions will only be published in the next issue.

SUMMARIZED

- Give the article a title.
- Include author's name, at least for internal use.
- Insert paragraphs.
- Please do not use self-created bullets, but use the tabulator.
- Divide long texts so that they can be spread over several issues if necessary.
- Write the article the way you would like to read it.
- Do not use RAW images but JPG.
- Respect the editorial deadline.

FINALLY

Unfortunately, I have already had to reject texts. But these are extremely rare exceptions. Either they were of such length that they went beyond the scope of the issue, or they were written in a style that made them difficult to read.

Finally, it should be said that I am always very uncomfortable rejecting texts. A lot of work usually goes into each article. I know that, after all, I write enough of them myself. In fact, I write texts myself that are proofread by other writers. I also don't like it when that editor, to whom I've given a story, rewrites my text and deletes passages. I have experienced that the story has lost reading flow, but in fact it becomes better in most cases.

I myself do not delete anything or very little in the texts. I make stylistic corrections with the utmost caution. Usually, however, I leave the text as it is, since it loses originality and authenticity when I delete or reword.

My concern is to finish the layout work of an issue quickly in order to get it out as close to the editorial deadline as possible. If I have to make a lot of corrections to an article, everything gets pushed back, plus I don't enjoy it anymore. After all, it's a hobby, not a profession; but in the end, I enjoy doing it. If the authors stick to the suggestions I've made, I'll keep having fun, too.

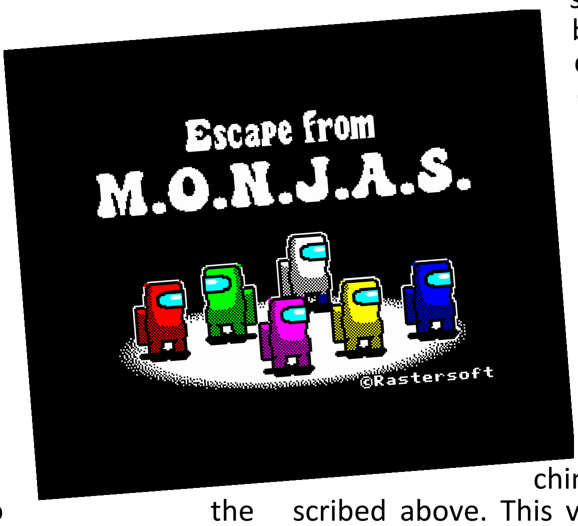
In this sense!
Your Redax Joachim

HOW TO HACK A GAME: MONJAS

BY THOMAS EBERLE

Hello everybody!

My last article on this topic must have been well received, so this time I want to describe an actual hack again. Since I'm creating the SCENE+ disk, I have to dive a bit deeper into the topic for this purpose, but this is a relatively simple hack. I want to show you how easy it can be and encourage you to try it yourself.



The background: If you use a Divide or similar, you can simply download TAP files from the internet and play them. But if you still use a normal Spectrum with floppy interface, you have to rewrite the programs to load them from floppy. This is not always easy, because you have to consider the limitations of the different floppy systems.

The easiest and also always my first step is the transfer to the *MB02*. Since I have a SD card interface on the *MB02*, I can simply plug the SD card into the PC and transfer a program.

But it's not that easy, you need a software that can handle the *MS-DOS* format of the PC and the *BS-DOS* format of the *MB02*. As described elsewhere, I use Total Commander with the appropriate plugins for MBH (MB02 hard disk format) and (not necessary here) *.TAP*. With the TAP plugin I can look at a tap file, similar to a zipped file. As you would expect, the program consists of Basic Loader, a load image and a machine code part:

```
 MONJAS - Basic 65 bytes
 screen.tap - loading screen 6912
bytes
 monjas_bin - machine code part
 code address 23990, length 41540 bytes
```

Let's look at the Basic first:

```
 1 PAPER NOT PI: INK NOT PI:
 BORDER NOT PI: CLEAR VAL "23989":
 LOAD "" SCREEN$:
 PRINT AT VAL "8", NOT PI;:
 LOAD "" CODE:
 PAUSE NOT PI:
 RANDOMIZE USR VAL "23990"
```

The basic is already designed to save memory, there is not much RAM left between BASIC and the start of the machine code. The RAMTOP is with 23990 relatively high. If you have a low number, it is high for the RAMTOP - that's what they say, not everybody has to understand it.

So this value with **CLEAR 23989** defines from where the Basic ends and the machine code can start. Between system variables and this RAMTOP is quite little space, but we will come to that problem.

NOT PI is a calculation formula and means: 0 (zero). So the first three commands are simply to set both the foreground (font), background and border to color 0 (black). **NOT PI** uses less memory than writing a number. The RAMTOP command »**CLEAR**« then determines where the machine code can start as described above. This value is also a mathematical formula, so in fact **VAL** »**23989**« uses less memory than writing **23989**, although at first glance it looks like more characters. The image is then loaded. The following PRINT command sets the PRINT pointer only to a certain position, so that the message does not destroy the loading screen when loading a program. For this there are other commands that disable the message, but probably again the most favorable solution for the memory space was chosen. Then the machine code part is loaded.

PAUSE NOT PI simply means a **PAUSE 0**, so wait for a keystroke. After that the machine code program starts.

The *MB02* disk interface is the perfect interface, because it emulates the cassette on another medium like floppy disk or here SD card. Nevertheless I always make the *MB02* versions in a way that it can be easily adapted for other systems. Of course, otherwise I would have the double work, because I always convert the programs for *MB02*, *+D/Disciple* and the *Opus Discovery*.

It is not obligatory with the *MB02*, but with other disc-interfaces, that a load command must always have a program name. Since I already know that memory is running out, I use short program names. I call the image M$, the code part Mc. But I don't want to save memory only with the Basic, also both program parts are packed. For the picture I use the utility »Screen Compressor«. It packs the picture in less than half of the bytes, so it uses only 3238 bytes. But the image is no longer loaded directly into the screen memory, but to address 50000 and called from there. So I have to add a **RANDOMIZE USR 50000**, again the basic part becomes longer.

I pack the code part with »Turbo Imploder«.

The inputs are quite simple, you specify the start of the MC, the length and the entry address, in this case the start of the MC and the entry address is the same. Turbo Imploder packs the whole thing in only 23041 bytes and no additional basic command is needed. Since the packer always needs some seconds to unpack, I remove the **PAUSE 0** command. I also delete the PRINT AT command, because disk load commands don't bring a "BYTES" message to the screen and so don't destroy the screen (but with the *MB02* it does, so it makes sense to put the command back in).

The BASIC now looks like this:

```
1 PAPER NOT PI: INK NOT PI:
BORDER NOT PI: CLEAR VAL "23989":
LOAD "M$" CODE:
RANDOMIZE USR VAL "5e4":
LOAD "Mc" CODE:
RANDOMIZE USR VAL "23990"
```

As you can see, I have also optimized the memory for the additional **RANDOMIZE** command. The value was set in **VAL ""** and **5e4** means nothing else than a 5 with 4 zeros, but it uses less characters and therefore less bytes. All in all this basic is even 5 bytes shorter than the original.

Done... or?

Of course not. A RAMTOP to 23990 is perhaps still feasible with the *MB02*, but *+D/Disciple* and *Opus* need more space in the RAM of the Spectrum. There the error message is preprogrammed. The solution: We pack the whole thing on address 25000. How now? Quite simple: With **LOAD "Mc" CODE 25000** the code is loaded to 25000 instead of the original address 23990. The **CLEAR** command is set to 24999... Enough space.

But does a program that was stored on address 23990 simply run on another address? Theoretically there is software which can do this, e.g. if only relative jump addresses are used. But in fact it is sure that it won't work, because there are always fixed jump addresses in the code and they won't work anymore. In addition, the code is packed, in the original it was 41540 bytes long and would have occupied almost the entire memory, too much to simply move.

The solution: We move the code again after everything is in memory and the disk interface is not used anymore. Then it doesn't matter if the system variables of the interface are overwritten. This simple assembler program puts the code back in the right place:

```
ORG 48042
LD HL, 25000
LD DE, 23990
LD BC, 23041
LDIR
JP 23990
```

Explanation: The ORG command sets the address at which this program is to start. In the HL register we load the address where the data to be copied is located, in the DE register we load the address to which the data is to be copied. BC contains the length in bytes to be copied. LDIR then executes the copy operation of the mentioned block, more precisely the content of HL is copied to the address of DE, then both values are incremented, the value of BC is decremented by one until BC is zero. At the end follows the jump to the program start.

I enter this assembler program into my assembler (PROMETHEUS in my case), assemble and save to 48042, which is the address after the program code of "Mc". I load both codes together and save them together:

```
LOAD "Mc" CODE 25000
LOAD "mon.cd" CODE
```
(this is the short new codefile)

```
SAVE "Mc" CODE 25000,23057
```
(code plus the short codefile)

Now change the BASIC, the entry address is no longer 23990, but 48042. A **CLEAR VAL "23989"** is inserted after the last load command.

Is it done now? Unfortunately not, a program start shows that the **CLEAR VAL "23989"** is not executed. The program has become too long by this **CLEAR** command, it does not fit *under* the RAMTOP anymore. But I can shorten actually nothing more. I could shorten the program names to one letter, but on a disk with about 20 programs I don't want to start that. The alphabet has only 24 letters, so I soon run out of them. I then split the BASIC. After the **RANDMIZE USR VAL "5e4"** I added another LOAD command : **LOAD "Mon.b"**

I packed the load and start command of the MC into another BASIC - program and saved it with SAVE "Mon.b" LINE 1.

Et voilà, I have now the following files:

- MONJAS - Basic 51 bytes
- M$ - Load image as codefile on address 50.000 with 3238 bytes
- Mon.b - Second basic file with 32 bytes
- Mc - Codefile on address 25000 with 23057 bytes

In total this is 26378 bytes from formerly 48517 bytes, almost half.

The Basic has become a little longer in total, but it has been split into two files:

MONJAS:

```
1 PAPER NOT PI: INK NOT PI:
BORDER NOT PI: CLEAR VAL "2499":
LOAD "M$" CODE:
RANDOMIZE USR VAL "5e4":
LOAD "Mon.b"
```

Mon.b:

```
1 LOAD "Mc" CODE:
CLEAR VAL "23989":
RANDOMIZE USR VAL "48042"
```

That's it. One more program on the floppy disk, you can see the whole thing on the
SCENE+ floppy disk. By the way a very funny game.

VARIABLES ON A THEME

FROM YOUR SPECTRUM ISSUE 2&5, MARCH AND APRIL 1984

System variables are bytes in memory which help the Spectrum remember certain things it needs to know about itself- if you like, the housekeeping routines. Delve deeper into the Spectrum ROM with Dilwyn Jones in this, the first of two articles which investigate the complete available set of system variables, giving comprehensive guidelines as to what you can and can't do with them.

The system variables are bytes in the Spectrum ROM which allow the computer to know all the things it should do to operate in the way you've come to expect. The information, such as how the Spectrum's memory is laid out, is held in the system variables in these addresses so that the computer can get hold of it and update it as and when required.

We can make use of the information stored in these memory locations in our programs, either by reading information already there or changing it to make the computer do something it might not otherwise do, or sometimes do it more easily.

Not all of them are that much use to us. And certainly not all of them ought to be changed. Some will cause the computer to crash, or the computer may simply ignore you. Some can be happily changed under certain circumstances only, and most within strict limitations. I hope to give you some guidelines as to what can and can't be done, but hopefully you will learn your own little **PEEKs** and **POKEs** in time as well.

23552 TO 23559 KSTATE READING THE KEYBOARD

When the processor is interrupted (50 times every second in the UK normally) one of the things done is to read the keyboard and store the results here. The bytes have different uses. Not all can be practically used by the programmer. You can use this program to examine what's going on in the eight bytes of **KSTATE**. Run it and press various keys to see what effect individual keys have, such as the Shift keys, and what effect going from one key to another has.

```
10 FOR A=23552 TO 23559
20 POKE 23692,0: REM KEEP SCROLLING
30 LET B=PEEK A
40 PRINT A; TAB 10;B; TAB 20;CHR$ B
   AND B>31
50 NEXT A
60 GO TO 10
```

The first four bytes of **KSTATE** deal with something called 'two key rollover' which allows you to press a second key before you actually let go of the first. The descriptions given to the main four bytes, 23556 to 23559, will apply to the first four also as long as you bear in mind that these only come into operation for two key rollover. **PEEK 23556** can return the code of the upper case version of the key pressed, so if you pressed Symbol Shift A you would get the code of 'A', not the code of 'a' nor the code of 'STOP'.

This may be useful where it is essential that upper case be entered, etc. The effect of pressing a key is temporary and lasts only as long as the key is being pressed. The value in 23556 would be 255 if no key was being pressed at the time the interrupt had occurred. For the Enter key, a value of 13 is returned. For the Space key, a value of 32 is returned.

Pressing both Shift keys simultaneously produces 14. This program will demonstrate this:

```
10 LET A=PEEK 23556
20 POKE 23692,0
30 PRINT A, CHR$ A AND A>31
40 GO TO 10
```

23557 is used for timing to prevent intermittent key contact, etc, causing problems - known as *keyboard debouncing*.

23558 is the auto repeat timer which times the pause before the keys start repeating, then the pause between repeats once the key has actually started repeating. The delays used are those in the system variables that hold these delays (23561/2).

23559 contains the code of the last character pressed on the keyboard. This depends on whether the Shift keys were pressed or not. The numbers produced are those that would be returned by **PRINT CODE INKEY$** except that these are the last key pressed and not necessarily the key currently being pressed. Try this program to display what can happen - RUN it and try pressing various keys making use of the Shift keys.

```
10 LET A=PEEK 23559
20 POKE 23692,0
30 PRINT A, CHR$ A AND A>31
40 GO TO 10
```

See also under 23611 FLAGS.

23560 LAST_K NEWLY PRESSED KEY

Every time the keyboard is scanned, a key is found to have been pressed and proved valid, the value of this system variable is updated. Its content is the code of the last key pressed.

This system variable does not really do much you could not do with **INKEY$**, except that it could be used to type ahead one character. If you try this program, you will find that if you press a key when invited to do so, the key is indicated on the screen in a short while even though the program may not have got as far as line 50 when you pressed a key. The code of the last key pressed is stored here and stays here until another key is pressed. It is possible to test for a newly pressed key by examining bit 5 of the system variable **FLAGS 23611**. This would be '1' for a key just pressed.

```
10 PRINT "Press a key now"
20 FOR A=1 TO 900
30 NEXT A
40 CLS
50 LET A=PEEK 23560
60 PRINT A: IF A>31 THEN PRINT CHR$ A
```

This could be used for testing for a y/n (yes or no) type situation - if you knew one was coming up, you could indicate your response before the program got there and the program would respond when it got round to it. Also, if two keys were pressed simultaneously, the program would respond if one were released without having to wait for the keyboard to be released completely.

Control characters can be generated using Caps Shift in conjunction with the number keys. Enter returns 13. Pressing both Shift keys together returns 14. To see this, try this program:

```
10 LET A=PEEK 23560
20 PRINT A, CHR$ A AND A>31
30 GO TO 10
```

23561 REPDEL REPEAT DELAY

This system variable contains the length of time that a key must be held down before it starts to auto-repeat. The time delay in the UK is one-fiftieth of a second and starts off at 35/50 of a second. You can happily **POKE** this if, for instance, you want the key to start repeating immediately. The cursors become rather difficult to control if you, say, **POKE 23561,1**. **POKE 23561,0** effectively turns off the auto-repeat, actually giving a delay of about five seconds like **POKE 23561,255**.

23562 REPPER DELAY BETWEEN REPEATS

This system variable controls the length of time between repeats once the auto- repeat has actually begun. The time is in fiftieths of a second in the UK. If you effectively want to turn off the auto-repeat for any reason, **POKE 23562,0** or **POKE 23562,255** gives about five seconds between repeats. If you wish to edit long program lines (eg. a long **PRINT** statement) then **POKE 23562,1** will speed up moving the cursor to the right place. But beware of changing 23562 too much at the same time or you may speed up the cursor so much it becomes difficult to control. Its normal value is 5/50 to a second or one tenth of a second.

23563/4 DEFADD

The address of the argument of a user- defined function in a program; ie. if you had **DEF FN A(B)** in a program line, the value in 23563/4 would be the address of the letter B in the brackets in that line while only the function is being used. The best way to **PEEK** into **23563/4** to show this is to put the **PEEK** as a part of the **FN** to be evaluated as there is always a zero there unless the function is being evaluated. So the line:

```
10 PRINT PEEK 23563+256*PEEK 23564
```

would always return zero. On the other hand:

```
10 DEF FN A(B)=PEEK 23563 +
   256*PEEK 23564
20 PRINT FN A(999)
```

would return the address of the B in line 10. The 999 is not significant, just something to actually give a value to B to prevent an error. In the case of a function with no argument:

```
10 DEF FN A()=PEEK 23563 +
   256*PEEK 23564
20 PRINT FN A()
```

this would print the address of the closed bracket symbol.

23568 TO 23605 STRMS

The first 14 bytes on a basic Spectrum contain addresses relating to channels and streams. Streams -3 to +3 are stored in two bytes each.

23606/7 CHARS

This system variable has as normal values:

23606 contains **0**
23607 contains **60**

This system variable points to the start of the character set that the computer uses for printing on the screen and the printer. **SCREEN$** also uses this system variable. The normal address pointed to is 15360 which is 256 less then the address of the start of the ROM character set. (256 less because the character generator is accessed by something similar to **PEEK 23606 + 256 * PEEK 23607 + CODE "A" * 8** and since the first character is a space, and the code of a space is 32 you can see that 8 * 32 is 256). The character generator is 768 bytes long, so if you wish to set up a new character set you must set aside this number of bytes in case it is overwritten by Basic - you wouldn't get a crash, you'd just end up with gibberish.

Mention was made of **SCREEN$** using this system variable - in fact, you may be aware of the problem that **SCREEN$** does not recognise user-defined graphics normally, unless they happen to be similar to an existing Spectrum character. In fact, **SCREEN$** works by picking up the address of the start of the character generator and looking through the table until it finds a matching character. Now since the Spectrum screen is bit-mapped rather than memory-mapped like some computers, once anything is printed on the screen it stays the same even if you change the character in memory. So we could temporarily change the pointer to the character set to point to the user-defined graphics and look up there.

One snag is that although there is a system variable that tells us where the user-defined graphics start, this address is 256 or more - so we must subtract 256. This conveniently means we subtract one from the high byte. This program should demonstrate:

```
10 FOR X=144 TO 164
20 PRINT AT 0,0; CHR$ X
30 POKE 23606, PEEK 23675
40 POKE 23607, PEEK 23676-1
50 PRINT AT 20,0; SCREEN$ (0,0)
60 PAUSE 40
70 POKE 23606,0
80 POKE 23607,60
90 NEXT X
```

What we did was make the computer think the user-defined graphics are the normal character set. **SCREEN$** will still produce characters with codes of 32-127, although this is easily overcome with a bit of fiddling. Since **SCREEN$** starts off with **CHR$ 32** and the UDGs start off at 144, we would need to add 112 to return characters in the range of the user-defined graphics.

Here is one way to do this. X is the x co-ordinate across the screen and Y is the y co-ordinate down the screen of the location **SCREEN$** is to examine. A check is first of all made that **SCREEN$** does not find one of the normal characters there, then returns if one is found. The character at Y,X is returned in A$. Line 8025 is needed only if you are using a character set other than the ROM one. If you are using the ROM character set, then delete line 8025 and replace lines 8070 and 8080 with the alternative versions that follow.

```
8000 REM SCREEN$ FOR UDGs
8010 LET A$=SCREEN$ (Y,X)
8020 IF A$()="" THEN RETURN
8025 LET A=PEEK 23606:
     LET B=PEEK 23607
8030 POKE 23606, PEEK 23675
8040 POKE 23607, PEEK 23676-1
8050 LET A$=SCREEN$ (Y,X)
8060 IF A$()<>"" THEN
     LET A$=CHR$ (CODE A$+112)
8070 POKE 23606,A
8080 POKE 23607,B
8090 RETURN

8070 POKE 23606,0
8080 POKE 23607,60
```

The story does not finish there. There are only 21 user-defined graphics - if **SCREEN$** does not find a match, it will continue looking up past the user-defined graphics until it has finished looking for the 32 to 127 range it thinks it's looking for. This could be embarrassing if there just happened to be some data stored above the UDGs for any reason which resembled any character. To help prevent this happening, although the UDGs are normally at the top of RAM anyway, this line could be added:

```
8065 IF A$>CHR$ 164 THEN LET A$=""
```

Incidentally, you should ensure that 23606/7 always points to the right character set when **PRINTing**, **LISTing**, etc, is done.

23608 RASP

Controls the duration of the buzz that sounds to warn you that you are running out of memory. At switch-on, this has a value of 64. This can be altered, but there seems little point. **POKE 23608,0** gives a very short click rather than a buzz - useful if you hate the buzz that satirically mocks you when you run out of memory. Alternatively, **POKE 23608,255** gives a very long buzz which immobilises the keyboard, preventing you typing any further than when the buzz sounded.

23609 PIP

Controls the length of the click emanated when a key is pressed in command mode or during an **INPUT**. It starts off at zero but may be changed. Any value between 30 and about 130 gives a pleasant, more audible bleep rather than the quieter click normally given. Values high than 130 tend to noticeably slow down the keyboard response (since computing stops as the bleep is sounded). Usually, the one to use is **POKE 23609,100.**

23610 ERR_NR

Controls error report number and normally has a value of 255 unless an error arises, when it contains one less than the error report codes printed, eg. for error 4, out of memory, it would contain a three. The message printed out is contained in ROM starting at address 5010 decimal. The end of the message is signified by the last character in the message having bit 7 set to a one. After the error message comes the '© 1982 Sinclair Research Ltd.' message that you see after switch-on or **NEW**. You can **POKE 23610** to generate an error to stop the program, but since the message printed is fixed and in ROM, you may end up with garbage.

If you wanted to simulate an 'out of memory' error you would end a program with **POKE 23610,3**. This would not work as any program line; you'd have to ensure that it was the last line, as once a condition arises to end a program, only then is 23610 looked at to determine what is printed.

23611 FLAGS

This system variable contains various flags controlling the Basic system and generally should not be **POKEd**. However, some of the flags can be usefully **PEEKed**.

Bit 0: Being a one indicates no space to be printed before the next keyword.

Bit 1: This bit being set to a one indicates that the print output is to be send to the printer. A zero would mean it is to be sent to the TV screen.

Bit 5: Any newly pressed key is indicated by its code being stored in 23560 (the **LASTK** system variable) and bit 5 of 23611 (**FLAGS**) is set to indicate that a new key has been pressed.

Bit 7: Syntax flag.

These will be of more use when using ROM routines in a machine code program.

23613/4 ERR_SP

Keeps track of the address on the machine stack where the appropriate return data lies. Try calling a few GOSUBs with no matching RETURNs and watch this point down the memory. Now you can see what happens and why this occurs when you run out of memory in a situation like this. Also, try PEEKing the contents of the three addresses (the base of which is pointed to by 23613/4) to see what return data actually consists of.

```
 10 LET A=PEEK 23613 + 255*PEEK 23614
 20 PRINT PEEK A; TAB 10; PEEK (A+1);
TAB 20; PEEK (A+2)
```

23617 MODE

Specifies cursor. Values zero, one, two or four specify the L/C mode, E mode, G mode or K mode respectively. **POKEing** this system variable will affect the appearance of the cursor - it may appear as a flashing letter, number, symbol or even a keyword. This is most apparent during an INPUT statement. The value is reset when the need arises, eg. a mode change made normally from the keyboard. So if you get into difficulties, press both Shift keys for E mode and then the same again to get back to normal L/C mode.

Try this program which **POKEs** all possible values into 23617. Most are variants on the four cursors, ie. you will find yourself in a particular mode after the POKE, such as everything coming as graphics as in G mode. 252 will give an L/C mode flashing '<' to point to where you're typing ...

```
 10 FOR A=0 TO 255
 20 PRINT A
 30 POKE 23617,A
 40 INPUT A$
 50 NEXT A
```

BORDER colour. Take a look at Table 1.

By **POKEing** various values into this system variable you could achieve a flashing, bright, multicoloured lower screen, or make both **PAPER** and **INK** the same colour to prevent other people getting at your programs - however, any alteration would have to be made blind. You could also make **INPUTs** extra bright to stand out.

BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
Lower screen FLASH	lower screen BRIGHT	BORDER colour und lower screen PAPER			lower screen INK		

23618/9 NEWPPC AND 23620 NSPPC

23618/9 is a two-byte system variable containing the line number of the line to be jumped to. 23618 contains the lower byte of the line number and 23619 the higher, so the line number contained is read as **PEEK 23618 + 256 * PEEK 23619**. To **POKE** a line number in, say line X:

```
 POKE 23618, X-256*INT (X/256)
 POKE 23619, INT (X/256)
```

We now come to system variable 23620. With 23618/9 and 23620, we could actually simulate a GO TO command to a statement within a program line, should that ever be necessary. **GO TOs** cannot access individual statements within long program lines.

To jump to statement four in line X, first go through the motions described above then **POKE** 23620,4 and the jump is executed.

23624 BORDCR

The bits of this system variable control the attributes of the lower screen and the BORDER colour. Take a look at Table 1.

By **POKEing** various values into this system variable you could achieve a flashing, bright, multicoloured lower screen, or make both **PAPER** and **INK** the same colour to prevent other people getting at your programs - however, any alteration would have to be made blind. You could also make INPUTs extra bright to stand out.

23627/8 VARS

The pointer to the start of the variables store. Apart from finding your way into the variables area, you can find the length of the Basic program with this expression. This excludes screen, system variables, stacks and variables.

```
 LET bytes=PEEK 23567 + 256*PEEK
23628 - PEEK 23635 - 256*PEEK 23636
```

23629/30 DEST

The address of the variable when it is assigned to. If the variable had been set up before, it would point to the start of where it was stored in the variables area. If it was being defined for the first time, it would point to the address of the start of the name of the variable in the program, eg. in 10 **LET A = 5**, it would point to the address of the letter **A**.

It can also be used to find the memory address of a numeric variable, if you use something like **LET A=A** as in the following program:

```
 10 LET A=5
 20 LET A=A
 30 PRINT PEEK 23629 + 256*PEEK 23630
```

23631/2 CHANS

Stores the address of where the channel information area starts.

23633/4 CURCHL

The address of **INPUT/OUTPUT** information used at that moment. It normally points during an **INPUT/ OUTPUT** operation to a five byte block of data in the channel information area. Use this program to examine the contents.

```
 1 FOR X=0 TO 3: PRINT #X; PEEK 23633
+ 256*PEEK 23634: NEXT X: PAUSE 0:
STOP
```

PART 2

In the last issue of Your Spectrum, an investigation was made of the system variables, 23552 through to 23634. Here we continue with a study of the remaining system variables, 23635 to 23689.

For the uninitiated, the system variables are bytes in the Spectrum ROM which tell the computer the things it needs to know to act in the way you've come to know and love. Information, such as how the memory map is laid out, is held in the system variables so that the computer can access it and update it as and when required.

23635/6 PROG

The address of the start of the area in memory where the Basic program is stored. This points to the first byte of the line number of the first program line. This may be useful if you're converting programs for other Sinclair Research computers with information held as a **REM** statement in the first line of a program. See also under VARS above.

If you wish to 'security lock' a line into a program, then by means of this system variable you could **POKE** a zero into both bytes of a line number at the start of a program. Program lines start with a two-byte line number.

23637/8 NXTLIN

The address of the start of the next program line. You could use this to enable you to access machine code stored within **REM** statements anywhere in the program, eg. those loaded with MERGE from a tape library of subroutines. These would have their own local calls to machine code like this:

```
9000 LET A=USR (PEEK 23637 +
     256*PEEK 23638+5)
9010 REM <> MACHINE CODE <>
9020 RETURN
```

One constraint to this is that you should not include any colour, flash, brightness, etc, control characters into the **REM** statement or they may be interpreted as machine code, upsetting things somewhat. However, if used from a library of subroutines, these would not normally be used anyway.

23639/40 DATADD

This contains the address of the comma ending the last item of data. If nothing was read from the list (eg. after **RUN** or restored, etc) the address held in 23639/40 is the address of the byte before the program area, normally the **CHR$ 128** at the end of the channel information area. To demonstrate try RUNing this program:

```
10 DATA "1","2","3","4","5"
20 LET A=PEEK 23639 + 256*PEEK 23640
30 PRINT A; TAB 9; PEEK A; TAB 18;
   CHR$ PEEK A AND PEEK A>31
40 READ B$
50 GO TO 20

23754 128
23763 44 ,
23767 44 ,
23771 44 ,
23775 44 ,
23779 13
```

The address in this two-byte system variable can point to the Enter character or the colon signifying the end of the line or statement containing the data - the address of the terminator of the last item of data.

23641/2 E_LINE

This system variable points to the start of the area above the variables. From this we can gain an idea of how much memory is used in bytes by screen, system variables, program and variables, once the program has been **RUN** to set up the variables, etc. Type this in, as a direct command:

```
PRINT PEEK 23641 + 256*
PEEK 23642-16384
```

We can also tell how much room is used for variables once the program has been **RUN** to set up the variables. Use the command:

```
PRINT PEEK 23641 + 256*
PEEK 23642 - PEEK 23627 -
256*PEEK 23628
```

23658 FLAGS2

This system variable contains some flags used (normally) by the computer to indicate certain conditions.

The best use we can make of this is to utilise the flag indicated by bit 3. This being a one indicates Caps Lock on or Caps Lock engaged.

What use is that? Consider in a program using **INKEY$**; eg. in a menu of options in a filing program, we often need to know whether the operator is pressing a certain key. If the operator is invited to press 'Y' for Yes or 'N' for No, they may press 'y' for Yes or 'n' for No - mixing up lower case and upper case capitals. Most often this would depend on whether Caps Lock was engaged - people are not interested in upper or lower case and whether they press 'y' or 'Y' they expect the computer to understand as humans would. But the computer doesn't really appreciate that. So if we engage Caps Lock ,automatically, our worries are over and we have a simpler program which doesn't have to check (as far as it's concerned) two separate options for each choice.

It is tempting to use the Basic statement **POKE 23658,8** to engage Caps Lock and **POKE 23658,0** to dis- engage it. But this will affect the other flags, so do check their state first unless you know they are not any particular value. Normally in L mode, 23658 has a value of zero so it is generally OK to use the **POKEs** above. You are not likely to cause crashes, but some funny effects may be caused in rare cases. When the printer buffer is empty, bit 1 will be zero.

23659 DF_SZ

This system variable contains the number of lines in the lower section of the screen, normally used for **INPUTs**, error reports and so on. Normally this would be a two, except for when a long **INPUT** prompt is displayed, etc. If a value of zero is **POKEd** in, normally to attempt to clear this unused part so that we can use the whole 24 lines of the screen, the computer crashes.

However, this can be done within a few restrictions. These are that we must ensure the lower part of the screen is restored to normal before any use is made of this - so to break out of a program would be somewhat catastrophic! Also, errors generated within the course of a program will have the same effect since the error report would have to be printed out.

Here is a short listing to demonstrate the use of line 22 and 23 on the screen. Unfortunately, it only works for **PRINT** or PRINT **TAB** as we cannot use PLOT down here and PRINT AT will only work down to line 22. The screen is restored to normal by **POKE 23659,2** within the program.

```
10 POKE 23659,0
20 FOR A=0 TO 23
30 PRINT A
40 NEXT A
50 PAUSE 0
60 POKE 23659,2
```

To demonstrate what can go wrong, let us generate an error by adding this line to the program:

```
45 PRINT error
```

Oops!!! If you just want to PRINT on the bottom two lines it is usually better to use **PRINT #1; "text"** which works just as well if not better, without such a risk of causing a system crash. If you POKE a value greater than two into **DF_SZ** the upper screen will become smaller than normal. So after **POKE 23569,Y** the upper screen would be 24-Y rows down and would scroll when the **PRINT** position got to or beyond 24-Y,0.

This program shows how a part screen scroll can be maintained with **DF_SZ** and **SCR_CT**. Here, random numbers appear and scroll up the top 14 lines of the screen only.

```
10 POKE 23692,0: POKE 23659,10
20 PRINT RND
30 GO TO 10
```

23670/1 SEED

When **RANDOMIZE** (number) is used, the number (a constant or a variable) is stored in this system variable. This is the number that determines the next random number. It opens up the possibility of cheating, since you could work out the next (supposedly) random number generated and use the knowledge gained to 'swing' luck your way. For example, after **RANDOMIZE 1**, the next value of **RND** would be **(0.0022735596**, **INT (RND *6) + 1)** to simulate a die being thrown as a one.

23672/3/4 FRAMES

This is a frame counter which can be used as a timer. It counts frames of a TV picture and so is incremented fifty times a second in the UK, or every 0.02 seconds, although the time taken to actually read and evaluate these three bytes of the timer may not allow it to be used to this accuracy. It has a timing range of nearly four days (actually about three days 21¾ hours). The manual (chapter 18) points out that you need to read the value of these three bytes twice in succession and take the high value for full accuracy because of the possibility of the values of the three changing while being read in such a way as to cause large inaccuracies.

It must be emphasised that the timer bytes are in the opposite order to what you might expect - the most significant byte is 23674, so the timer values are read by:

```
65636*PEEK 23674 + 256*
PEEK 23673 + PEEK 23672
```

which returns time in units of fiftieths of a second.

There are several things that affect the accuracy of this timer. Using **BEEP** stops the timer. Using the printer and loading/saving, etc, also affect its accuracy. However, the use of **PAUSE** is OK as this only waits a specified time without re-setting or stopping the timer.

23675/6 UDG

The address of the start of the dot patterns for the user-defined graphics is normally 32600 on a 16K Spectrum or 65368 on a 48K Spectrum. This number is the same as **USR "a"**, so **PRINT USR "a"** corresponds to:

```
PRINT PEEK 23675 + 256*PEEK 23676
```

Compulsive POKEers can have fun with this one. The manual suggests changing this to save space by having fewer user-defined graphics. However, it is also possible to do the reverse, and set up more than one user- defined graphics set if required; however, only one set of 21 can be in use at any one time. Remember that since there are 21 UDGs it is necessary to set aside 21*8 (168) bytes for each separate set of UDGs and **POKE** the start addresses, into 23675/6, of the character set in use.

For fun, type in the following commands:

```
POKE 23675,96: POKE 23676,127
```
(16K Spectrum)

```
POKE 23675,96: POKE 23676,255
```
(48K Spectrum)

Then, using the user-defined graphics (they normally appear as capital letters until re-defined) try to type out a message. I'll leave you to find out what happens.

One useful tip: once you've set up a user-defined character set it may be SAVEd on tape. Most people would type something like:

```
SAVE "chars" CODE 32600,168
```

Fine, but you have to specify the start addresses. You could use **SAVE "chars" CODE(PEEK 23675 + 256 * PEEK 23676), 168**, and then you could happily save the current set of UDGs on tape without knowing the address of where they start. This would, for example, allow you to LOAD a character set SAVEd from a 16K Spectrum back into a 48K Spectrum without having to know the addresses.

To get a character set back into the right place on a machine with different amounts of memory, simply use **LOAD "chars" CODE(PEEK 23675 + 256 * PEEK 23676),168**. This would automatically re-locate data to the right address for the machine in use at the time. This is the same as **LOAD "chars" CODE USR "a"**, which saves a bit of typing although it may look a bit strange.

23679 P_POSN

Contains information about how far across the printer buffer the **LPRINT** position has got to. Contains (33- column number) for columns 0 to 31.

23680 PR_CC

Contains the low byte of the address where the next character is to go into the printer buffer, ie. this will contain (23296 + **LPRINT** column numberer), being zero for the left column of the printer, 15 for the 15th column etc. Because this is the address of the top row of dots of each character, you can POKE this to change the **LPRINT** buffer position provided you change the value in **P_POSN** (**23679**) to match. It may appear to work if you don't do this, but problems will be encountered at the end of the line.

23681 UNUSED SYSTEM VARIABLE

This system variable, although strictly speaking unused, usually contains 91. This is the high byte of the **LPRINT** buffer address (91*256 is 23296 where the buffer starts). This can be POKEd for your own use but using the printer will overwrite it back again to 91. 23680/1 together contain the address of the **LPRINT** position in the printer buffer. You will not affect the working of the printer if you **POKE 23681**, but anything stored here may be over- written by the printer routines.

23677/8 COORDS

23677 is the system variable that contains the x co-ordinate of the last plotted point. After **CLS** it starts off at zero and 23678 is the system variable that contains the y co-ordinate of the last point plotted. It contains the actual value, so if the last point plotted was 3,3 both bytes would contain threes.

These two can be POKEd with valid x and y co-ordinates respectively. Since POKEing these does not actually PLOT anything on the screen, this is a convenient way to move the PLOT cursor around. This could be done by **PLOT OVER 1;X,Y: PLOT OVER 1;X,Y** - but would be messy. Amongst other things this could simulate **MOVE** found in other BASICs - useful if you wanted to draw lines from around a particular point.

23684/5 DF_CC

The address in the display file of the **PRINT** position. It may be POKEd to send the **PRINT** output elsewhere although this requires an understanding of the way the display file is organised.

23688/9 SPOSN

23688 contains information concerning how far across the screen the **PRINT** position has got. It starts off as 33 for the left-hand side of the screen and decreases by one every time the PRINT position moves one place to the right. After using **PRINT AT Y,X;** (if Y and X are valid **PRINT AT** co-ordinates) 23688 would contain 33-X. This can be useful when trying to prevent words being chopped in half when printed on the screen. If you imagine the number in 23688 as counting down towards zero as there is no more room on the current line, you can see that comparing this to the length of the word to be printed gives us an idea of whether it is necessary to move to a new line to prevent the word being chopped Suppose the word to be printed was **W$**:

```
IF PEEK 23688<LEN W$+1 THEN PRINT
```

This only works for words less than 32 characters long.

23689 contains information relating to how far down the screen the print position has got to. It starts off at 24 for the top line of the screen and decreases by one every time the **PRINT** position moves down the screen. If you do not want a scrolling display and would rather the screen was cleared when the **PRINT** position got near the bottom of the screen, then try:

```
IF PEEK 23689=3 THEN CLS
```

23693 ATTR_P

Contains permanent attributes, or the attributes (**FLASH**, **BRIGHT**, **PAPER** and **INK**) in effect globally. Local colours in PRINT statements, etc, are dealt with elsewhere. Note that most of the ROM routines use the values of the system variable holding the temporary attributes as these contain the permanent attributes unless a local parameter is specified. **CLS**, however, clears the screen to the colours, etc, in **ATTR_P**. The functions of the individual bits are shown in Table 1.

Bit 7 is one for **FLASH 1**.
Bit 7 is zero for **FLASH 0**.
Bit 6 is one for **BRIGHT 1**.
Bit 6 is zero for **BRIGHT 0**.
Bits 5, 4 and 3 contain the **PAPER** colour in binary, eg. for **PAPER 7**, bits 5, 4 and 3 would be 1, 1 and 1 respectively.
Bit 2, 1 and 0 contain the INK colour in binary, eg. for **INK 3**, bits 2, 1 and 0 would be 0, 1 and 1 respectively.

Attributes of eight or nine are not dealt with here. If the permanent attributes are eight or nine, then those stored in 23693 may not be valid.

23694 MASK_P

This is the system variable that helps the Spectrum determine the attributes of anything printed when a parameter of eight is specified. So, if you specified BRIGHT 8 globally, bit 6 of 23694 would be set to one to remind the computer in future that BRIGHT 8 has been specified. So, to determine the colour/ flashing/ brightness when printing, the computer looks at what's already there and prints the word in that colour, etc. Or if you like, it only overprints the character on the screen and leaves the attributes alone. You can see what each bit does by looking at Table 2.

Bit 7 is one when **FLASH 8** is in effect.
Bit 6 is one when **BRIGHT 8** is in effect.
Bits 5, 4 and 3 are normally all one when **PAPER 8** in effect, but see below.
Bits 2, 1 and 0 are normally all 1 when **INK 8** in effect, but see below.

When there is more than one bit to consider, as in **INK** and **PAPER**, then only the bits set have their attributes bit taken from the screen. This can lead to some unexpected effects. Try this:

```
10 INK 8
20 POKE 23694, BIN 00000011
30 PRINT AT 0,0; INK 5;"5555555"
40 PRINT AT 0,0; "1111"
```

As **INK 8** is specified, you may expect the ones to be printed in cyan like the fives, but no. Rather than check the **INK** attribute as a whole, it only checks the bits set in 23694, which were bits 0 and 1. See if you can work out what colour the ones will be printed in. Have fun!

23695 ATTR_T

This system variable contains the current temporary colours as would be set up by local statements within **PRINT** statements. You could see this for yourself with something like these two direct commands:

```
PRINT PEEK 23695
PRINT INK 7; PAPER 0; PEEK 23695
```

That is, include the **PEEK** in a **PRINT** statement under the effect of the local colour controls. Normally unless local colour statements are specified, this system variable will contain the global colour values. Colours, etc, to be used for printing on screen are taken from these temporary system variables and things are balanced such that **ATTR_T** is only different from **ATTR_P** if local colour attributes and so on so decree. The function of the individual bits can be seen in Table 3.

23696 MASK_T

This is rather like **MASK_P** (system variable 23694) except that the parameters here are temporary. Normally the same as the equivalent permanent parameter 8s, this is changed while local colour 8s, etc, are in effect. You could study this by using something like:

```
PRINT PEEK 23696, INK 8; PEEK 23696,
INK 0; FLASH 8; PEEK 23696
```

The individual bits have the functions illustrated in Table 4.

23697 P_FLAG

This system variable contains, as you might expect from its name, flags used during printing. After **PAPER 9** has been specified, bits 6 and 7 are set to one. After INK 9 has been specified, bits 4 and 5 are set to one. After **INVERSE 1** has been specified, bits 2 and 3 are set to one. And after **OVER 1** has been specified, bits 0 and 1 are set to one. The effects are global if the odd numbered bits (bits 1,3,5 and 7) are set to one, and temporary if the even bits (bits 0,2,4, and 6) are set to 1. See Table 5 to see what I mean.

23681 23728/9

These three bytes in the system variables are not normally used by the Spectrum - you may like to make use of them as 'custom variables' for use in your own programs in which you need to access information. These are particularly useful in machine code routines where you can simply access the information via an address rather than searching for the variable in the variables area. 23728/9 was intended for use by non-maskable interrupts but these don't occur on the bare Spectrum.

23730/1 RAMTOP

This two byte system variable points to the last byte of RAM of the Basic system area. Note that this is not the end of the memory used by Basic, in the sense that the user-defined graphics normally hide up above this address. If you move RAMTOP up above the start of the user-defined graphics they may be overwritten, but you gain quite a few valuable bytes which may be useful for 16K users.

One important thing is that **NEW** only operates as far as the address held in 23730/1, so you can store data above this which may be passed between programs loaded into the computer. The same is true if you want to preserve machine code routines, etc.

23732/3 P_RAMT

This contains the address of where RAM ends on the Spectrum. If you acquire a Spectrum whose memory capacity you don't know, then don't bother looking inside it to see if it's an expanded model or not, just enter this expression:

```
PRINT PEEK 23732 +
256*PEEK 23733 - 16384
```

The 16384 bytes subtracted is for the ROM since RAM starts at address 16384 and goes up to the address held in **P_RAMT**.

The tables alongside are referenced in the main text of the article, and show the functions of the individual bits of specified system variables: table 1 refers to 23693 ATTR_P; table 2 - 23694 MASK_P; table 3 - 23695 ATTR_T; table 4 - 23696 MASK_T; and table 5 - 23697 P_FLAG.

Tabelle 1: 23693 ATTR_P							
BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
FLASH	*BRIGHT*	*PAPER colour binär*			*INK colour binär*		

Tabelle 2: *23694 MASK_P*							
BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	
temp. FLASH	*temp. BRIGHT*	*temp. PAPER colour*			*temp. INK colour*		

Tabelle 3: *23695 ATTR_T*							
BIT7	**BIT6**	**BIT5**	**BIT4**	**BIT3**	**BIT2**	**BIT1**	**BIT0**
temp. FLASH8?	*temp. BRIGHT8?*	*temp. PAPER8?*			*temp. INK8?*		

Tabelle 4: *23696 MASK_T*							
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
FLASH8	*BRIGHT8*	*PAPER colour 8?*			*INK colour 8?*		

Tabelle 5: *23697 P_FLAG*							
BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT1
global PAPER9	*temp. PAPER9*	*global INK 9*	*temp. NK9*	*global INVERSE1*	*temp. INVERSE1*	*global OVER1*	*temp. OVER1*

This article is an extract from the book by Dilwyn Jones, Delving Deeper Into Your Spectrum Rom, first published in the UK by Interface Publications and costing £7.95.
The links to the original articles are as follows:.

Your Spectrum, Issue 2,
March 1984 - System Variables

`http://www.users.globalnet.co.uk/~jg27paw4/yr02/yr02_78.htm`

Your Spectrum, Issue 3,
May 1984 - System Variables

`http://www.users.globalnet.co.uk/~jg27paw4/yr03/yr03_97.htm`

FINALLY:

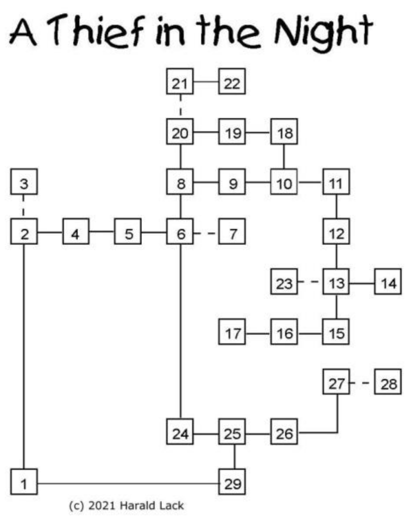
Here, independent of the previous article, are a few useful **POKEs**:

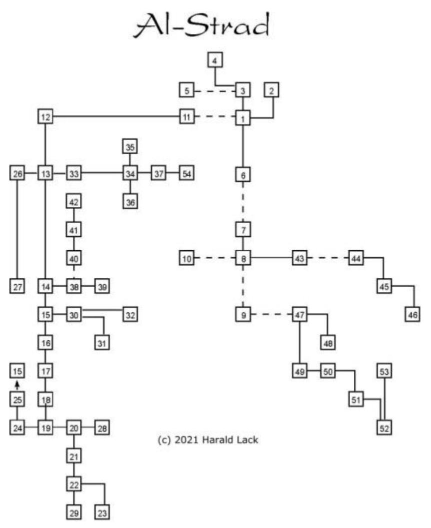
- Get total memory:
```
PRINT PEEK 23732 + PEEK 23733 * 256
16kB-Spectrum: 32767
48kB-Spectrum: 65535
```
- Determine the current free memory space during the programme run:
```
PRINT 65535-(PEEK 23653+256*PEEK 23654
```
- Scroll to 255 pages:
```
POKE 23692,255 or
POKE 23692,0
```
- Key tone:
```
POKE 23609,50
```
- **Cursor speed:**
```
POKE 23562.1
```
- Timing:
```
10 POKE 23672.0: POKE 23673.0
20 LET S=(PEEK 23672+256*
PEEK 23673)/50
```
- Set first basic line to 0:
```
LET a=PEEK 23637+256*PEEK 23638:
POKE a,0: POKE a+1,0: POKE 23756,x
```
Sets the first line in the programme to line number x. If x=0, the first line cannot be edited easily. Of course, the POKE command can also be used to undo the whole thing.
- Determine RAMTOP:
```
PRINT PEEK 23730+256*PEEK 23731
```
- Finished routines in ROM:
```
RANDOMIZE USR 3435: CLS
RANDOMIZE USR 3756: COPY
RANDOMIZE USR 6137: LIST at line 0
RANDOMIZE USR 3438: Deletes line 23 & 24
```

ADDENDUM TO ISSUES 229 & 230

Unfortunately the plans for the adventure solution of
Al-STRAD from issue 229 and A Thief in the Night from issue 230 got lost on the way - I simply forgot them. Please forgive me!

Here are the plans!





DUAL-ROM ON THE ZX SPECTRUM 128K+2

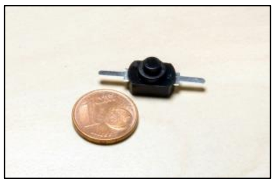
BY FRANK DOPIERALA

My last article ended with me still waiting for the parts to repair my microdrive, but in the meantime I had caught the completion fever of another "construction site": My "ZX SPECTRUM 128K+2 restoration project".

Basically, it was completely restored, but still not finished (ROM switching). that didn't leave me resting, so I preferred this project to the restoration of the microdrives. The spare parts for these microdrives did arrive in the meantime, so there's no need to hurry.

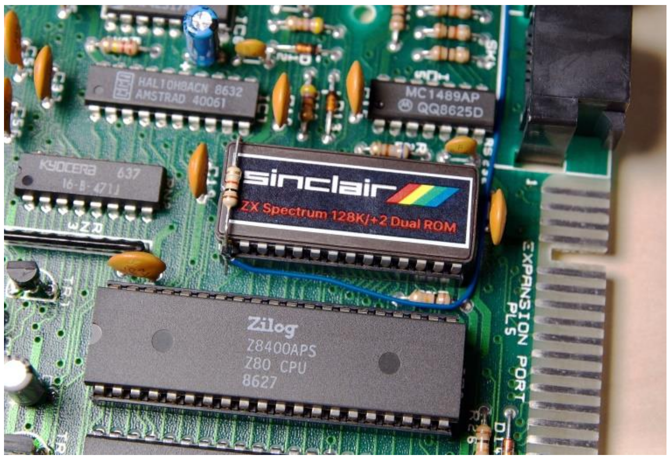
I had mentioned in my last article, that with such retro-work not only the result counts for me, but also the way and quality, i take my time! And since I am a fussy perfectionist, small details which some might consider superfluous, take me an eternity to complete. And they are only finished when the perfectionist in me has given his OK. One of the characteristics of my perfectionist is, that when it comes to conversions or technical additions of any kind, the conversion should not look nerdy and tinkered with, but rather as if the manufacturer himself had intended it to look that way. Technical elegance is important to me.

The original ROM in my ZX SPECTRUM 128K+2 is fortunately socketed. Otherwise I might have been tempted to simply piggyback another EPROM on top of the already soldered ROM, like I did in the past with my first Speccys. I also thought about using this technique anyway or to take a double sized EPROM to burn both ROM images, the one from the +2 and the one from the 128K TOASTRACK, into it. For this I first had to check if the EPROMS 27256 and 27512 are pin compatible. Fortunately they are, only the PIN for the programming voltage is somewhere else, but the JUNIOR-PROMMER recognizes this automatically. So it was clear to me that I will use a single 64K EPROM to accommodate both. However, at this moment I did not know how to burn both images together to one file. Again, the EPROM burning program PINATUBO for my ATARI TT helped me quite intuitively: I specified the 27512 EPROM as the medium to be burned and then simply load one of the two 32K files as the image to be used. PINATUBO then "complains" that the image is too small for the selected EPROM. But now I can select "Append" instead of "Load" under "File" and PINATUBO automatically appends the second 32K - done.



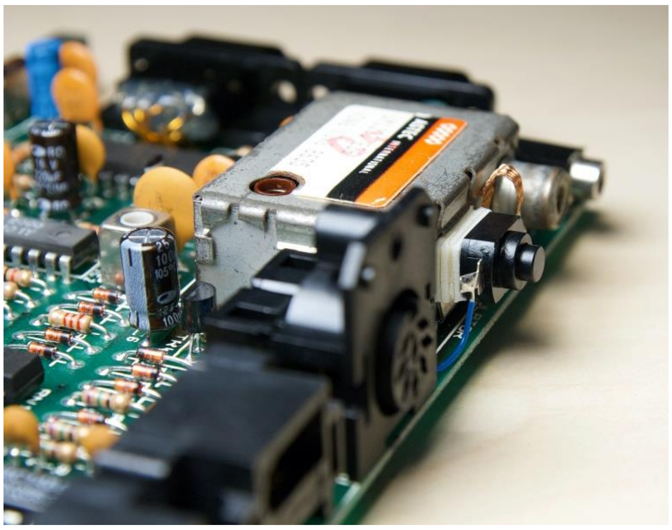
After the EPROM was burned, I exchanged it against the original ROM, without any further change. PIN 1 of the socket in the Spectrum is directly provided with +5V, so the "attached" ROM image had to be active. Switched on: All running well, reporing the Amstrad message. After that I attached the Diag card and checked the checksum: Everything OK! Next Step: Spectrum off, EPROM removed, PIN 1 of the EPROM bent out and connected with crocodile clip directly with ground: Sinclair message!!! Checked again with the Diag card, everything error-free! Now I did label the EPROM with an appropriate sticker, as UV protection and so that it looks pretty.

The next question now was, how to realize the ROM-switching, so it would look professional. First I thought of a small slide switch on the right side of the Spectrum, directly opposite the joystick ports. But for this I would need to lead at least two cables from the switch to the mainboard. And either I soldered these cables directly or I had to think about a plug connection, if I wanted to remove the mainboard. Both versions did not please me. So I took a closer look at the mainboard and



the bottom part of the case. There is one small gap between the case and the modulator. I googled for miniature switches and found them on Ebay. Ordered, measured and thrilled.

The switch was even so nicely rectangular that it fitted exactly between the base plate and the lid of the modulator. After I had marked the position of the switch with inserted mainboard exactly at the housing, the hole was drilled. I then taped the switch to the modulator with four layers of double-sided and elastic tape. The four layers emerged through experimentation: There had to be just enough layers to hold the switch against



the case wall with enough pressure when the mainboard is inserted, even when the switch is actuated. After that, I shortened the two solder tags and soldered a piece of desoldering litz against the modulator base plate on one of them, so that the switch remains movable when inserting the mainboard.

Since the switch only switches through ground, the EPROM at PIN 1 (address line A15 to ground = low = address range 0 to 32767, A15 to +5V = high = address range 32768 to 65535) must therefore be set to +5V by a pull-up resistor if the switch is open. Since the supply voltage for the EPROM is directly opposite at PIN 28, an easy way: Simply solder a 1K pull-up resistor across, done! In advance, I shortened a piece of switch wire to the exact length for the connection between switch and EPROM. I then soldered this to PIN 1 of the EPROM at a 45 degree downward angle. I then soldered the other end of the switch wire downwards towards the mainboard to the second solder tag of the switch. The course of the wire on the mainboard can be seen on the picture.

When the mainboard is now inserted, the switch finds its guide with the hole in the backplane quasi automatically and fits perfectly. The switch is designed to stick out a bit when closed and be almost flat with the case when open. Now for the final functional test, first with the switch closed (sticking out). Turn Spectrum off, push in the switch, turn it on again: Voila! I like the technique and the realization of the switch so far very much!

But, there is still a little bit missing, the missing labeling lets recognize the switch as a later installation. This i had to change with lettering foil, in this case by letters, which can be removed by water from the carrier foil and then well positioned applied. Now, after applying the two-line lettering, I like the overall result. NOW my restoration project »ZX SPECTRUM 128K +2« is finished and I can finally play one of my favorite games with the Speccy, which are »CYCLONE«, »HIGHWAY ENCOUNTER«, »WORSE THINGS HAPPEN AT SEA« ...

SHARP MZ 700 AND MZ 800

BY JOACHIM GEUPEL

Today I would like to introduce you to two more computer classics from my collection. Both of them I experienced by myself when they came on the market during my job in a computer store. They are the computers **MZ-700** and **MZ-800** from **Sharp**. Both are very similar in design and almost identical in features - except for a few basic features.

The computers could have been groundbreaking, but had gone into wayside like so many other computers released in the early 80s.

HAYAKAWA METALS

Many will suspect, others actually know: SHARP once made faucets, belt buckles for belts without holes, and of course mechanical pencils. Let's be honest! Who didn't know that?

Sharp was founded in 1912 by **Mr. Tokuji Hayakawa**, shortly after he invented and patented the said belt buckle. **Mr. Hayakawa** was quite an ingenious inventor, sort of the Japanese Clive Sinclair - only he was much more successful in the long run. The workshop he founded was called **Hayakawa Metals**. His mechanical pencil, the **Hayakawa Mechanical Pencil**, was a great success and was patented beyond Japan, even abroad. In which city the predecessor company of **Sharp** was founded is not quite clear. What is clear, however, is the company motto that Mr. Hayakawa based his products on: *»Make products that other companies want to imitate«,* which probably accounts for his success.

After the *Great Kantō Earthquake*, which destroyed large parts of Tokyo and the port city of Yokohama, the company relocated to Osaka. In 1925, it produced the first crystal detector, the simplest equipment for receiving radio broadcasts, which was followed by the first tube radio in 1929. From 1931, the company was involved in the research and development of television sets, making it one of the first companies in the world to tackle the radio reception of moving image transmissions. In 1935 the company went to stock exchange and in 1942 renamed itself to **Hayakawa Electric**.

SHARP

From 1953 the company became the first Japanese company with mass-production of televisions. **Sharp's** first television set, the **TV3-14T**, cost 175,000 yen - at that time, the starting salary for school graduates was 5,400 yen per month. The set thus easily cost 32 months' salary. At about the same time, **Hayakawa Electric** changed its company name to **Sharp**. The name was derived from the name of the popular mechanical pencil, the **Ever-Ready Sharp Pencil**, called **Ever-Sharp Pencil** for short. The **Hayakawa Mechanical Mencil** was patented and marketed as the **Screw Pencil** or **Popelling pencil**.

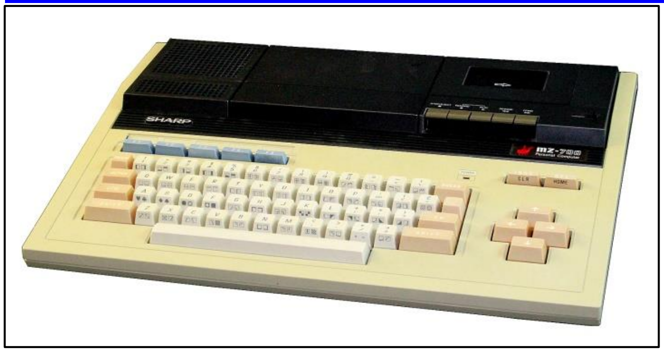
Quite early, in 1963, **Sharp** developed the first transistor-based desktop calculator **Sharp Compet CS-10A**, a desk calculator with a Nixie tube display, which was released in 1964.

The development of calculators and computers continues through to the late 1980s. In 1978 **Sharp** released the **MZ80K**, one of the first true desktop computers that were freely programmable. The **MZ80K** was in direct competition with the **APPLE II**, the **TRS-80** from **Radio Shack** and the **PET** from **Commodore**. At about the same time as the **MZ80A**, the **Sharp MZ 3500** was released, which was intended more for the commercial sector and looked quite similar to an **IBM PC**. The **MZ80K** was followed by the **MZ80A** and as its direct successor the **MZ-700**.

Parallel to the desktop and home computers, Sharp made a name for itself with the pocket computers. What today is a "Personal Computer", called a PC, was also a PC in the 80s, but at this time PC stood for **Pocket Computer** In 1983, the pocket computer PC 1245 was introduced to the people, followed a year later by the **PC1246**. Two of these Pocket Computers are in my collection, which I will present sometime later. The pocket computer series ended with the **Sharp PC 2500**, which was rather one of the early laptops. The **HC-4100** was already a handheld computer with LCD display and **Windows CE 2.0**in 1997.

My personal memories of the two MZ- computers refer to the time at the end of 1984 and the beginning of 1985. When I was working in a computer store in Reutlingen in the last two months of 1984, the two computers were real stars in this store. The boss of the store was short and fat, had an idea of money but only a rudimentary idea of computers. The store was full of all kinds of computers. There were **Epson QX-10** computers running **CP/M**, there were **Oric's**, the obligatory **C64**, the first **CPC-Computer** and one or two **ZX Spectrum**. His main sale were with the **Alphatronic PC's**, where he had the 5 ¼-inch floppy drives replaced by the then hypermodern 3 1/2 -inch floppy drives. While the **TA-PC8** was set up relatively unspectacularly in a corner, the two **Sharp** computers stood as eye-catchers in the entrance area. Everyone who entered the store inevitably stumbled over the computers. While the display of the **MZ-800** shone with colorful, high-resolution graphics, the drawing pens of the plotter had to be replaced every two days on the **MZ-731**, because customers of my age and the somewhat younger ones liked to hog the plotter to put demo graphics and their own works of art on paper. All the computers were of no use to my boss; a year later he was bankrupt and later went to jail for a year for embezzlement and multiple fraudulent bankruptcy. I for myself was no longer affected by this. I had a viral flu at the turn of the year 84/85 and was in bed with fever. While I was sick, he kicked me out. I found another job in an electronics company, and exactly one year after my dismissial I participated in the forced sale of the remaining inventory from my old boss...

THE SHARP MZ-700



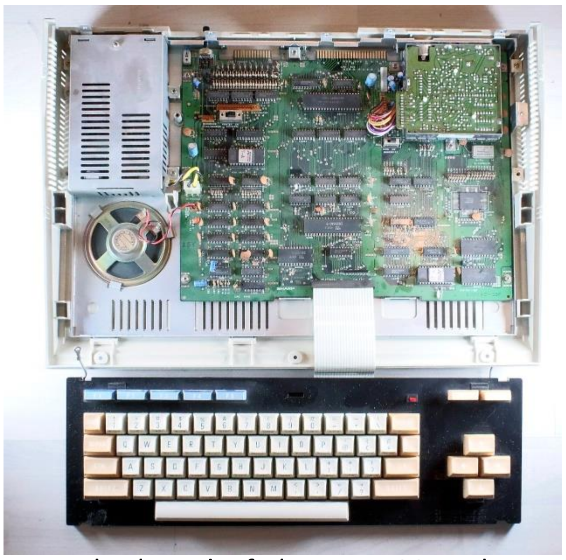
The **Sharp MZ-700**is, just like its predecessor **MZ80K** and its successor **MZ-800** a so-called CLEAN COMPUTER. By this **Sharp** meant that the computer had only a 4kB ROM, which contained an assembler monitor, input/output routines for the monitor and the speaker and a loading routine for the cassette recorder. The obligatory Basic was loaded from cassette, which had the disadvantage of having to wait several minutes for it to start. The advantage was that any available programming language could be loaded. There were a lot of languages adapted for this machine. There were more than five versions of BASIC, several assembler, Pascal, Lisp, C, Fortran, Comal (a synthesis of Basic and Pacal), Forth and some others.

The computer was delivered in four different configuration levels. The basic model was the **MZ-711**, which had only the most necessary basic equipment. Basically, it is possible to connect an RGB monitor, as well as a normal video monitor and the obligatory TV set. A Centronics-compatible printer port, whose connection was led out similar to the Edge Connector of the **Sinclair ZX-Computer**, is also present. Two 3.5 mm jack sockets allow loading and saving of programs. Two joystick ports, which are led out as five-pin headers, are not compatible with anything. It is not possible to connect an Atari compatible joystick without some effort.

The **MZ-721** has a built-in cassette recorder located at the right side. The ear- and mic-connectors on the backside are not replaced by the cassette recorder, but cannot be used anymore.

The **MZ-731** has a four-color plotter in the center, which can be used to plot graphics and drawings, as well as to output listings, which of course takes time.

In all models, a **Quick-Drive** can be used instead of the cassette drive. The **Quick-Drive** is a floppy disk drive for 2.8 inch floppy disks. They are not compatible with anything. Instead of being written in tracks, as is common with other disk formats, the disks are written in a continuous spiral, similar to vinyl record. Both the **Quick-Drive** and the 2.8-inch floppy disks are very rare to buy - and if they are, they are exorbitantly expensive.



The motherboard of the **MZ-700** makes a very tidy impression. Everything is solidly manufactured and clearly arranged in the case. The loudspeaker has a diameter of 77mm and makes a decent noise, the power supply is a compact switching power supply and the video modulator provides a composite video connection and a RGB socket in addition to the obligatory RF output.

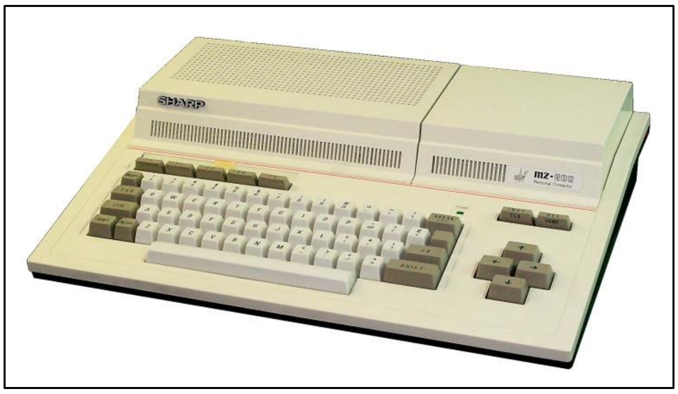
Sharp has given the computer a customer chip. Both the memory controller and the CRT controller are contained in a single customized LSI chip, the M60719. It combines on itself the 8x8 dot generator, which outputs 40 characters and 25 lines to the screen. The font displayed depends on the 4 kB ROM in which the characters are stored. It manages the monitor ROM, the DRAM, the video RAM, and the peripherals keyboard, timer, etc. which are assigned to the memory. It generates the clock for the Z80A processor and selects the printer I/O port. This IC is small but powerful.

The processor is a Z80A apparently built under license by SHARP with the designation LH0080A. So that there is no doubt, the processor designation is printed next to it.

There are two obvious slide switches on the board. One of them is used to switch between the built-in and the external cassette recorder. The second switch selects between built-in plotter and external printer.

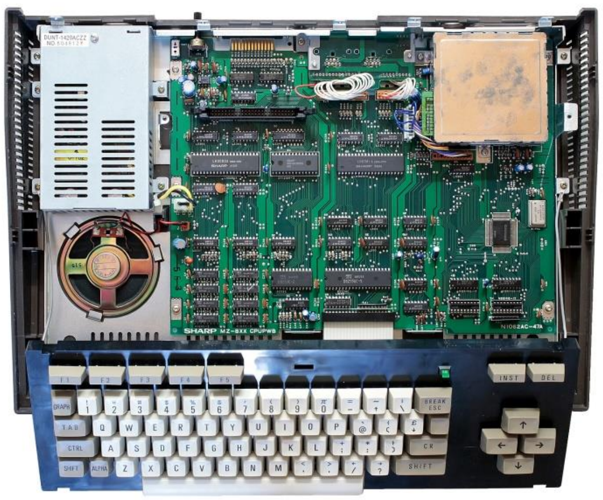
The computer was offered for about 800 DM, whereby the price quotations, which can be found on the Internet, fluctuate strongly. One site says 248,000 yen, which is about 3000 DM, another site says 510 DM. I think I know that the computer store sold it for said 800 DM (ca 400 Euro).

THE SHARP MZ-800

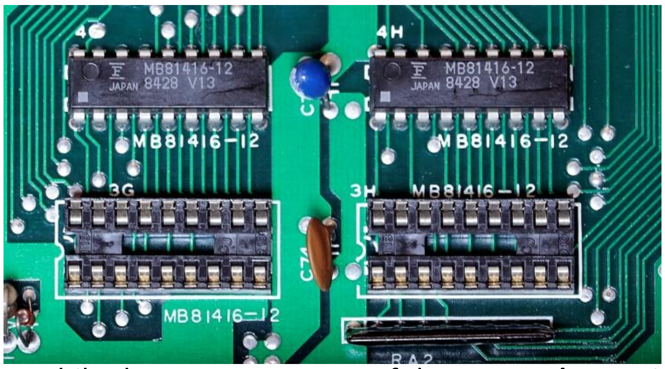


Like the **MZ-700**, the **MZ-800** is also a **Clean Computer**. It is the successor of the MZ-700 series. It is similarly equipped as the **MZ-700**, but can be upgraded to 128 kB.

Sharp has also divided the individual models according to their equipment. At first impression, it looks a bit more bulky than its predecessor. The basic unit, the **MZ-811**, has neither a cassette drive nor a plotter, and the **MZ-821** version still has a cassette drive integrated. The connections are the same as in the **MZ-700**: RF and composite output, RGB, joysticks and a Centronics port. Two expansion slots are hidden behind a cover.



The computer already differs externally from its predecessor. The upper part of the case is in a continuous creamy white. The keys show in a dark grey, in the **MZ-700** they show themselves in a more or less beautiful ocher.



While the upper rear part of the **MZ-700's** case is still divided into three parts to accommodate the cassette drive and the plotter, the **MZ-800** no longer has an internal plotter. Instead, this space is taken up by two slots into which interfaces such as a floppy disk drive or a RAM disk can be plugged.

A Z80 processor also works in the **MZ-800**. In the **MZ-700** it is clocked with 3.5MHz, in the **MZ-800** with 3.55MHz. In text display, the **MZ-700** was still content with 40 characters in 25 lines, the **MZ-800** has two text modes, namely switchable from 40 to 80 characters per line with the same number of lines.

Graphically, the **MZ-800** is better equipped. While the **MZ-700** only has seven colors, the **MZ-800** can display 16 colors. The number of pixels on the **MZ-800** is 320 x 200 or 640 x 200, which is much larger than on its little brother, which only has 80 x 50 graphic dots in correspondingly blocks.

The motherboard, like that of the **MZ-700**, is nicely tidy and clearly laid out. **SHARP'S** own LSI chip for memory management and CRT controller is also found here again. Eye-catching is the wide connector into which the ribbon cable for the two expansion slots is plugged.

Two empty IC sockets can be found on the board, into which two 16kB x 4 bit memory IC's can still be plugged. It can be assumed that the monitor ram can be upgraded by plugging in appropriate ICs.

Various descriptions of the computer state, that it can be upgraded to 128 kB. There are no slots for additional RAMs, except for the mentioned IC sockets on the board. So a second assumption of mine is that the memory upgrade to 128kB is in the form of a Expansion board that slides into one of the two free slots.

All in all, the **MZ-800** is a very nice computer. It is very well made, easy to disassemble and reassemble, and just looks good. It is an advantage to have a cassette drive installed. Without this data recorder it is difficult to load programs, because the computer does not want to work properly with several external recorders.

SUMMARY

Both computers are fun to use when they work. They are solidly built and look good. However, if Sharp's own LSI-Chip, M6071 is defective, they are only suitable as a decorative element in an exhibition with the reference that this was once a computer.

BASIC GIMMICK

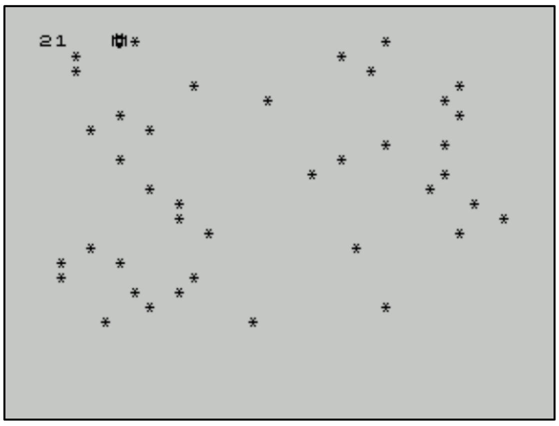
A little game. The mini spaceship has to avoid the stars. It is controlled with the »a« and »l« keys. The game can be interrupted with »q«.

A counter runs along with the game and displays the score at the end.

At the end, you are asked whether you want to restart the game. Pressing y restarts the game, any other key ends the game. There is no doubt that the game can be improved.

Have fun with it.

```
 10 FOR a=0 TO 7
 20 READ b
 30 POKE USR "A"+a,b (graphic "A")
 40 NEXT a
 50 DATA 36,189,255,165,165,189,189,24
 60 LET na=5
 70 LET pu=0
 80 CLS
 90 PAUSE 5:
100 PRINT AT 20,RND*31;"*"
110 PRINT AT 20,RND*31;"*"
120 PRINT
130 POKE 23692,-1
140 PRINT
150 LET pu=pu+1
160 IF  SCREEN$ (0,na)="*"
    THEN GO TO 250
170 PRINT AT 0,0;pu
180 LET a$=INKEY$:
    IF a$="a" THEN LET na=na-1:
    IF na=-1 THEN LET na=0
190 IF a$="q" THEN STOP
200 IF a$="l" THEN LET na=na+1
210 IF na=32 THEN LET na=31
220 PRINT AT 0,na;"\a" (graphic "A")
230 GO TO 90
240 REM
250 PRINT AT 0,0; FLASH 1;
    "POINTS ";pu
260 FOR a=0 TO 50
270 BEEP .01,RND*40
280 NEXT a
290 PRINT #0;"Again (y/n)"
300 PAUSE 0
310 IF INKEY$="y" THEN RUN
```



ADVENTURE SOLUTION »CALLING«

BY HARALD LACK

The Calling (of the Demon)
1987 Terry Taylor and F. J. Neary (Visual Dimensions)

Dear fellow users!

It's hard to believe what you can find when you search your whole collection. I found the adventure game "Calling" mentioned above, which has surely been hiding in my dark boxes for too long. It was written in 1987 by TERRY TAYLOR and F. J. NEARY using the Professional ADVENTURE WRITERS and published by VISUAL DIMENSIONS. The program is available in both 48K and 128K versions, whereas my solution refers to the version for the SPECTRUM 48K. I don't know if there are differences in content between the two versions.

The program offers the possibility to use RAM SAVE and RAM LOAD as well as to switch pictures on or off. This makes it a bit more practical, you don't have to handle so much with external storage media.

So I thought I'd give it a try... which isn't that easy and took me a few months of pondering.

The program itself belongs for me basically to the good adventures, the directions and also the location descriptions are meaningful. One thing is a bit annoying, the program does push you for action. When you want to read a text carefully (which can make sense in adventure games, because often helpful tips and hints are hidden there), you are always asked to do something, the text of the location sometimes disappears too early, before you have read it completely. If in doubt, just type "look" and the text is there again.

It is noticeable that the program contains a lot of objects, which rarely contribute to the solution. This must have been a little joke of the programmers.

In my opinion, the biggest obstacle to the solution is the part where we are in our study room and have used the CHARM, because after that there is a SAY CONFIDENCE, which is not necessarily immediately obvious (see hints below). So much in advance.

Here's the short background story to the program: The player is on the way back from York with his girlfriend Jenny , where we had dinner, across the Yorkshire Moor towards Bayley, our home, when our good car lets us down in the middle of the country. In addition, the distant thunder portends a storm. Well, this could happen. Jenny immediately offers to look for help and is gone before we know it.

In a nearby house she tries to call the towing service. Unfortunately, there are no more signs of life from her, so that after two hours of waiting we go in search of her, since we suspect that something has happened to her, especially since this strange property is alone in the middle of nowhere. In addition, we still need the tow truck, since our vehicle will certainly not move from the spot by itself. So we basically know what our task will be.

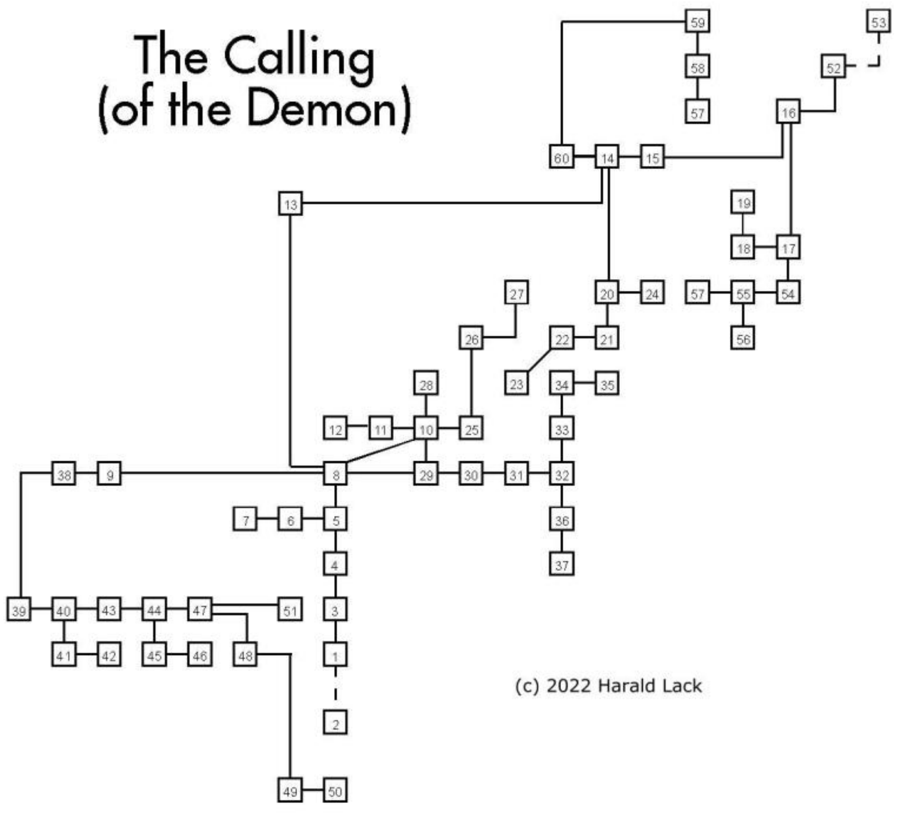
Of course I am aware that many players like to solve an adventure by themselves. Therefore, before my step-by-step solution (I hope I have always transferred my handwritten notes correctly, which is often an adventure in itself with my writing), i will provide four hints that may help even if you do not want to use the overall solution.

- You can only carry one key at a time. Since there are two of them (RUSTY KEY and SKELETON KEY) in the program, it makes sense to put them in the right place.
- In this adventure it makes sense to talk to one or the other person to get valuable hints. The parser understands also relatively complex sentences. However, it is important to use quotation marks (" ") when addressing a person, for example in the form: Say to x "follow me".
- In the course of the adventure we will come across a useful copper helmet. It will serve us well. However, it is important not to wear it outside the building, because it attracts lightning, which is not good for us.
- When we find Jenny, she will follow us. Unfortunately, she is a bit weakened and lags behind us. That's why it's especially important to wait for her at the end, because her phone call will help us to solve the game.

So much for the four tips I would like to put at the beginning. Thus everyone can go itself on the solution way. Following now the altogether 60 locations of my plan with its included objects (so far they are useful / needed for the solution):

01)	Standing in an old country lane
02)	Within the broken down red car / pair of gloves
03)	Walking along a dark pavel driveway
04)	Standing outside a large imposing house
05)	Standing in the middle of a vast entrance hall / some armour
06)	In a large dusty room

07) In a spacious music room / grand piano, flute
08) Walking along the main corridor
09) In the old study / oak desk, diary, hole in the west wall
10) In the kitchen of the house / kitchen table, carving knife
11) Walking in an east-west corridor
12) In the drably decorated servants quarters / music magazine
13) Standing upon the highly ornate main staircase
14) Standing on the rickety landing
15) Standing in the base room of the north-east tower
16) Standing upon the cold stone floor of the north-easttower´s lower room
17) Standing under a sloping roof at the east end of the large attic
18) In the middle of the large attic room / picture on the north wall
19) In the previously hidden room / rusty key
20) Striding along the upper corridor A
21) Striding along the upper corridor B
22) Within the walls of a highly decorated bedroom
23) In a long low room / metal sphere
24) Standing in the upper gallery / valuable painting, skeleton key
25) In the neatly arranged kitchen garden / some garlick, some hemlock
26) Standing under the branches of a tree
27) Perched among the damp leaves of the tree / crystal ring
28) Among the smartly scrubbed shelves of the pantry / cooked ham
29) Entering the dining hall of the house
30) Standing on the damp paving stones of the terrace
31) In the middle of the beautiful gardens
32) Strinding along the damp gravel path that runs north-south
33) Standing upon a gravel path
34) Standing in a large grassy hollow
35) Inside a large kennel-like building / some junk, golden charm
36) On a gravel path in the rain-soaked garden
37) Standing within the confines of the gardener´s lodge / sturdy chest, piece of paper
38) In a gloomy alcove
39) At the bottom of the spiral stairs
40) At the west end of the long cellar corridor
41) In the low-roofed living quarters / pair if overalls
42) Standing in a small but well equiped kitchen / chocolate biscuit
43) Walking along the east-west cellar corridor
44) Strolling along the east-west corridor that runs the full length of the cellar
45) Standing in Quinn´s strange laboratory
46) In the corner of the cluttered laboratory / copper helmet
47) At the east end of the cellar corridor
48) Walking along the dark flight of stairs
49) Deep in the bowels of the earth
50) Standing below the ceiling of a vast subterranean tomb / some robes, large tomb
51) Standing in the dimly lit robing chamber / some boots
52) Standing at the top of the north-east tower
53) Clinging pecariously to a platform / strange machine, Quinn, Jenny
54) In the attic library
55) Walking along an attic corridor
56) In a small but used attic study / small desk, telephone
57) Stading in a part of the attic that had been made into a store
58) Standing under the sloping roof to the west of the large attic
59) Within the confines of the north-west tower´s lower room
60) Standing in the base room of the north-west tower



That was all about the plan and the items. For all those who don't want to try it themselves (yet) it goes on like this...

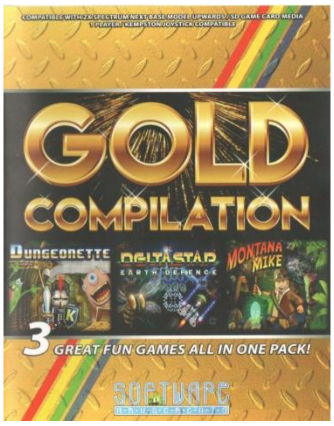
(We start in a country lane with a violent storm in a broken car), INVENTORY (we wear a suit - we come from eating), EN-TER CAR, GET GLOVES, EXIT CAR, N, N (now outside the said house), N (the entrance hall), EXAMINE ARMOUR, W, W (the music room - hopefully well soundproofed), GET FLUTE, PLAY FLUTE (we need a music paper first), E, E, N, W (the study room), READ DIARY (it belongs to Professor Quinn), E, NE (the kitchen), W, W (the servants' area), GET music MAGAZINE, EXAMINE music MAGAZINE, E, E, SW, U (stairway), U (on the landing), E, U, S, W (the middle of the attic), EXAMINE PICTURE (represents a human ear), PLAY FLUTE (due to our clumsy flute playing, the picture changes, finally dissolves and reveals a previously hidden passage), N (the absolutely necessary secret room - what would adventures be without it?), GET RUSTY KEY (the magic contained in it burns our fingers and we prefer to drop it), WEAR GLOVES, GET RUSTY KEY (already better), S, DROP FLUTE, DROP MAGAZINE,

E, N, D (at the foundation of the northeast tower), W, S, S, W (a bedroom), OPEN DOOR, SW, GET SPHE-RE, EXAMINE SPHERE (this will be our light source in the future), NE, E, N, N, D, D, DROP RUSTY KEY (I already mentioned the one with the keys above - it can only be one - Highlander principle), U, U, S, E (the upper gallery), EXAMINE valuable PAINTING (an old man riding a horse), EXAMINE PAINTING CAREFULLY (there is the second key of this adventure), GET skeleton KEY, EXAMINE KEY, W, N, D, NE (again the kitchen), N (the chef won't let us into the pantry), E (the kitchen's vegetable garden), GET GARLIC, GET HEMLOCK, EXAMINE HEMLOCK (toxic! !!), N (at the foot of a tree), U (on the tree), GET crystal RING, EXAMINE RING, D, S, W (back to the kitchen), GIVE GARLIC TO COOK (this is exactly what the chef is fixated on), N (the pantry), GET HAM, EXAMINE HAM (we discover a small cut on the side that could be from a knife), PUT HEMLOCK IN HAM (and already the ham is poisoned), S (back to the kitchen), GET KNIFE, EXAMINE KNIFE (surprisingly sharp), S, E (onto the terrace), E (and on into the garden), E (a gravel path), N, N (a grassy hollow), E (in the gutter, here's a scaly beast that's lusting after the ham; since it's poisoned, it doesn't do it any good and it gradually dissolves - comes from being so greedy), EXAMINE JUNK CAREFULLY (we find a golden zau-ber item), GET CHARM, EXAMINE CHARM (there are magic symbols on it),W, S, S, S (we reach the gardener's hut, who is not very happy about our appearance, because he looks very angry - at this point we can take only one action, so don't waste time), KILL GARDENER WITH KNIFE, DROP KNIFE, EXAMINE CHEST (has no lid), LOOK IN CHEST, GET PAPER FROM CHEST, READ PAPER (contains some clues for us), N, N, W, W, W, W (the main course), W (the study again), EXAMINE OAK DESK, INSERT CHARM (into the existing hole in the wall - fuses with the wall and leaves it in perfect condition), SAY 'CONFIDENCE' (as said above, by no means forget the quotation marks; a loud breaking sound is heard and a passage appears in the west wall), W (in the alcove), WEAR RING, SAY TO SPHERE 'LIGHT' (you absolutely must have the ring on or it won't work - the glow begins), D (the inevitable basement), E, S (the living area), GET OVERALLS, EXAMINE OVERALLS, WEAR OVERALLS, N, E (to the kitchen), GET BISCUIT, EXAMINE BISCUIT, DROP BISCUIT, W, N, E, UNLOCK STEEL DOOR WITH SKELETON KEY, S (Professor Quinn's lab), E (an interesting corner in the lab), GET HELMET, EXAMINE HELMET (only suitable for "home use" - see my comments above), WEAR HELMET, W, N, E, D, D (the entrance to the tomb where a very old ghost tries to attack us, but we have the helmet to protect us from it), E (into the tomb), GET ROBES, EXAMINE ROBES, W, U, E (an invisible force field prevents us from progressing), WEAR ROBES, E (this is the robe chamber in the temple of Danrath), GET BOOTS, EXAMINE BOOTS, W, W, W, W, U, DROP SPHERE, REMOVE HELMET, DROP HELMET, REMOVE ROBES, DROP ROBES, DROP PAPER, E, E, GET RUSTY KEY, U, U (at the landing), E, U (the northeast tower), U (to the top of the tower - we shouldn't miss the hole in the ceiling), WEAR BOOTS, JUMP (we jump through the hole in the ceiling thanks to our boots and land on a quite strange platform... Professor Quinn is also here but when he sees us he gets scared, slips and falls), EXAMINE MACHINE, UNLOCK MACHINE WITH RUSTY KEY (Jenny is locked inside and we can free her), D (Jenny follows us, though a bit slowly), D, S, S, W, S, EXAMINE TELEPHONE, WAIT (we wait until Jenny is there and calls the towing service - then she wants to go back to the car), N, W, N, N, D, E, D (the central staircase), D, S, S, S, S (on the country road), WAIT (until finally the tow truck comes and takes us away). And already we are rewarded with the final message which reads:

CONGRATULATIONS
You are on your way home and have completed the game.
Well done! You have scored 100%.

I don't want to add anything more to that. I had a lot of fun working my way through the adventure. I hope I could give some valuable tips.

SPECTRUM NEXT GAMES REVIEW
»GOLD COMPILATION«
BY THOMAS EBERLE


COMPATIBLE WITH ZX SPECTRUM NEXT BASE MODEL UPWARDS · SD GAME CARD MEDIA
1 PLAYER · KEMPSTON JOYSTICK COMPATIBLE
GOLD
COMPILATION
DUNGEONETTE
DELTASTAR EARTH DEFENCE
MONTANA MIKE
3 GREAT FUN GAMES ALL IN ONE PACK!
SOFTWARE

With the so-called **»GOLD COMPILATION«** I received three commercial games for the **SPECTRUM NEXT,** which I would like to present here. Most of you should already know the titles, but since not everyone follows the current events, here is a summary:

The **COMPILATION** consists of three titles of the company **SOFTWARE** AMUSEMENT. This company has already published **several** titles, also for modern systems like Android or Amazon Fire TV. These three titles are also the first titles of the company for the **SPECTRUM** NEXT and after booting the SD card in the NEXT, the three titles are displayed in a menu with sound:

1. **DUNGEONETTE**
2. **DELTASTAR EARTH DEFENSE**
3. **MONTANA MIKE**

DUNGEONETTE

The whole game reminds of **Atic Atac** and this is probably no coincidence.

You control a knight through a maze, looking for keys to open doors and find treasures. The knight has a sword, but can only throw small knives. The enemies found in the maze are destroyed by several knife hits, some not at all. If you are touched by an enemy, you can't move for a moment, which usually causes another hit, and you get energy drained. You can replenish the energy by eating.

The original **Atic Atac** game was once a success in 1985; this may still evoke memories today, but not really inspire. The game has no sounds at all and the graphics only stand out from the average Spectrum game because of the colors, but not because of the richness of detail. All in all, this is certainly no reason to have a **NEXT**.

DELTASTAR EARTH DEFENSE

This game is a kind of **Galaxian** clone. You control a spaceship at the bottom of the screen, and above you formations of aliens, some of which swoop down, firing from all guns. The goal of the game is to shoot down all the aliens, then the next screen appears. Variety is provided by the different types of aliens, as well as some bonus boxes that also fell down from above. You can shoot the boxes and they change some effects that ultimately occurs when you collect them. Here you can get new weapons, extra lives or bonus points.

It's not exactly innovative to remake a game from 1979. While there is some nice space background graphics, not much else has changed. There is no music in the game, only in the title menu a soundtrack welcomes us, which reminds me of my own first attempts with AY music. Still, it's fun to play for a short while in between.

MONTANA MIKE

For those who remember Rick Dangerous, this is a successor to **NEXT**. Who doesn't know it: Basically it's a platformer with a hero in the style of Indiana Jones. His basic weapon is a whip, but he can also find and use other weapons. You wander through a pyramid from screen to screen, trying to find the exit in each frame without losing a life. The hero can hop, climb and crawl, at least crawling does not appear in every platformer.

The game has decent graphics, at least by Spectrum standards. You could certainly get more out of the **NEXT**, but it's not bad. There is a theme song in the main menu, but no music in the game - only FX effects.

All in all, the compilation is a transfer of well-known game principles to a new device. After all, **SOFTWARE AMUSEMENT** was one of the first semi-professional distributors to take on the NEXT, even before the devices were shipped to most of the users. The SD card with the games comes in a nice box, but there is no manual for the games. Unfortunately, at the first try the games didn't run at all, after inserting the SD card and booting, I got a confused picture. Only after some research I found out that this game activates the SCANDOUBLER, but since I connected my Next via a RGB cable, the confused picture came out. A little hint here: press NMI and then "2" , that turns the Scandoubler off again.

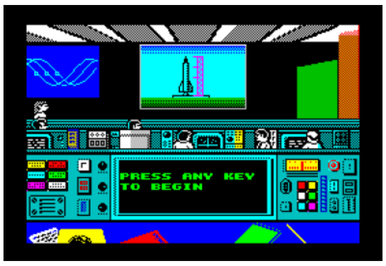
The sale of the games (and others from SOFTWARE AMUSEMENT) ended on 28.02.2021 and since then the games are freely available. Here is the link:

`https://itch.io/s/27215/zx-spectrum-next-games-bundle`

GAMES 2019/2020
BY ELLVIS

Slowly we are moving towards our goal to have reviewed almost all very good games until the actual date. Here are the reviews up to year 2020:

»AD LUNAM PLUS«
© 2019 / 2020 BY ALESSANDRO GRUSSU



I wasn't around when mankind landed on the Moon for the very first time, but I can imagine the fascination by a whole space race during the 50's and 60's. And as computers of that era helped the landing, Spectrum can help me to go through the space race by myself. And even more, to be a part of it!

AD LUNAM PLUS is an enhanced version of turn-based strategy game about the development of all the equipment and then using it to land on the Moon. The original game had few rough edges, but the enhanced version is worth playing it.

At the beginning we choose the country - USA or USSR. The game is trying to be accurate and that help to reach a bit of authenticity. You will need some time at games start to set the numerous options. Every choice in the game is made by selecting an option using numbers, sometimes we have to input a value (like a number of researchers we want to dedicate to a project). There is no time limit to make a choice, once we are fine, the last option is to end our turn. Computer play the other country at the background and we get reports of it after each turn is finished. This is in a form of short messages, like reports from secret services, changes in government funding and so on. It is keeping us updated in a picture so we don't feel playing alone. Also, one turn represent 3 months, so we have 4 turns per year.

Complexity of the game is quite big. We have to keep an eye on research and development of the space equipment (like moving researchers between the projects), recruiting and training the astronauts, building facilities but also planning missions. As game goes on, things we have to take care about each turn is much more then at the beginning. And that is where the strategy is, we want to be the first people on the Moon, do we?

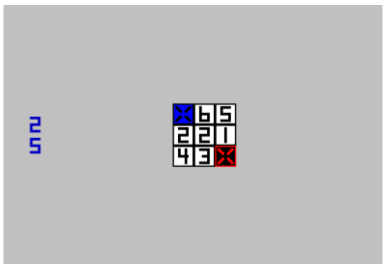
Quite often we will see messages like "Mission Failed". Don't let it that to put you off, it is normal that things doesn't work perfectly from the beginning. It is part of the game and it help to decide where to focus. Just a few more tries and you are up in the space.

The game is for 128K SPECTRUMS only. It does not contain a great amount of sound, but there is really nice graphics to guide us through the game. The concept and controls are very easy and it need not much time to learn what is what and how to play. There is also a SAVE option, so we don't have to play it straight from start to end as it will take some time. If you like strategy games at least a bit, give **AD LUNAM PLUS** a try, you won't be dissapointed!

Download at:

`https://spectrumcomputing.co.uk/entry/36138/ZX-Spectrum/Ad_Lunam_Plus`

»CARDINAL CHAINS«
©2020 VON COMPILER-SOFTWARE (MIGUEL ANGEL GARCIA PRADA)



Last issue we had a closer look on **COLORISTIC**, a small logical game. Well, **CARDINAL CHAINS** is basically the same thing with a small twist. We still have to fill all the playing field by one or more colours (dependin on level). The difference is, that each cell contain a number. We can go from lower number to the higher one, but we cannot go opposite, from the higher to the lower number. And this is the main difference between the two games. Everything else is very similar, we have minimal screen design, this time we get no music.

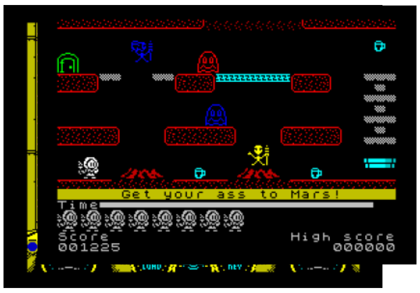
There are no passwords, but we can choose which level to start the game with »P« (this is right direction with QAOP Keys) in the menu. Because the numbers, I find this game a little bit easier then **COLORISTIC**. On the other hand, here we have 313 levels to go, so I am sure I will stick with this game for some time. And I am sure that if you liked **COLORISTIC**, you will find this game well worth playing too!

Download at:

`https://spectrumcomputing.co.uk/entry/36456/ZX-Spectrum/Cardinal_Chains`

»COLOCO«
©2020 TUXEDO GAMES (MANU SEGURA)

And we are on the rescue mission once again. But now without guns and without hordes of enemies. Sound boring? Well, I wouldn't say so...



Coloco is an interesting Version of the **Thrust** type of games. We are trying to save astronauts scattered around the planet's underground and underground's facilities. Because our rocket is small, we can save just one person per level.

First levels look easy. Just one screen, we start on one side, getting the astronaut on the other side of the screen and then flying back. But once you start you'll get why this game is not too easy. There is inertia force involved in the controls, so the flying is quite difficult. It need's a couple of tries to get used to it. But let's take a closer look. There are 3 types of platforms we can land, yellow are the ones we have to get the astronauts to, blue are where astronauts are and green are occasionally scattered around the levels, usually with some helpful object on it. We cannot land anywhere else besides those platforms. To make things a bit easier, landing can be done from quite high, so we don't have to be too careful with the landing itself.

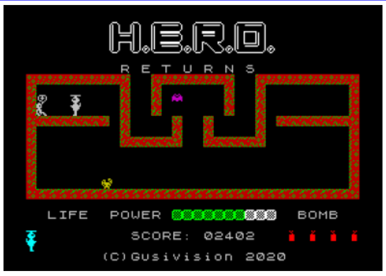
As we go through the levels, we have to fly over more screens. Also enemies start to show up, from mines to floating aliens. To make things even more difficult, astronauts are being kept behind the locked doors, so we have to get a key first, but flying around cost us a lot of fuel. Luckily, there are also spots with fuel to be pick up. Sometimes we can even get a bonus life. This can be picked up every game so at the end some levels can be completed for many tries without actually losing a life. At the beginning of the game we get 15 lives to go, which may seem plenty, but later levels will proove us differently. Under the playing field, we see an information panel with number of lives (left), current load (red when our space ship is empty and green when we got an astronaut), key (red when we don't have any and green when we have one) and finally a fuel status. Game is using **Churrera Engine MK1** that is well know from many and many platform games (I am sure you will be familiar with the sound effects the game is using) and this game show how versatile the tool is. The graphics are very nice and clean, we always see what is going on. The sounds are only for Spectrum 48k.

Coloco is an interesting game and if you are bored by all the platformers and want some kind of different action, gives it a try. You may very well like it.

Download at:

`https://spectrumcomputing.co.uk/entry/36342/ZX-Spectrum/Coloco`

»H.E.R.O. RETURNS«
© 2020 VON GUSMANB



Being in age doesn't always mean only drinking a beer in a front of TV. Especially if you've been a hero in the past and you see that people still need you. Time to shake the dust off your equipment and rescue some people.

H.E.R.O. used to be one of the top games back in the early 80's. The game was about a hero (you) who is equipped with a flying suit and a few bombs going down the caves and rescuing people. **H.E.R.O**. was fun to play and in 2020 we've got a homage to it : **H.E.R.O.** returns!

The game starts at the surface with an entrance to the cave. Once we enter it, we have to reach it's end where the person to rescue is sitting. Good thing is that the way is usually straightforward and there is not really a place to get lost. To compensate that the game make things complicated by placing us into narrow corridors where we cannot touch ground and ceiling. Such passages are really difficult to pass.

The caves are not too big, but also they are not empty. Once a while we meet a bat or spider, crab or other creature. Some of those can be killed by our small laser gun. We also have bombs, but those have no effects on the enemies at all and can be used only to break walls. Besides the enemies, I mentioned already some surfaces that cannot be touched by our hero. Those are lava - bright red/yellow stone and water. When flying above the water, we have to be careful about aligators showing up once a while.

The game look quite different from the original. The graphics are more detailed, but everything is smaller. Also, water is just a flashing texture, but original had nice waves. But most important is the gameplay and that is still pretty good. A nice help is that once we lost all our lives (we get 4 at the game start), we are allowed to continue with another 4 lives at the screen we died.

H.E.R.O. Returns is a nice little game that pay homage to the golden era of gaming. If you have a couple of minutes and you like games with a a bit frustrating gameplay you should try it.

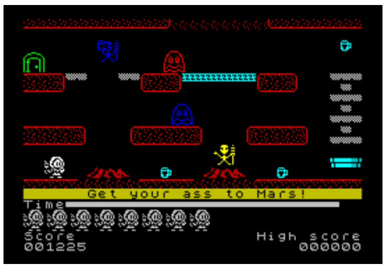
Download at:

`https://quantumsheep.itch.io/jumpin-jupiter`

»JUMPING JUPITER«
©2020 VON QUANTUM SHEEP

I would also like to address at least one platform game. My cup of tea is nearly empty and this game is exactly what I need - collecting tea cups all around the solar system. So, let's go!

Jumping Jupiter is a classic platform game in the **Manic Minner** tradition. Each of the 35 levels is



just one screen where you have to collect all the cups and sometimes also a key. When that is done, you enter the exit and go to the next level. To make it more difficult, there are various enemies we have to avoid.

Nice feature of the game is that you can use just one hand to control the main character (her name is **Jupiter**) using O, P and M. The other hand stays out of action and can, for example, keep a cup of tea or something. One good thing is that we can fall from any height without losing a life (with **Manic Miner** you therefore lost a life). Also, during the jumps (and falls) we can still control the direction and this is a great help in many situations. Although the speed of the main character is not the highest ever, it add to the action and the game playability is still very good.

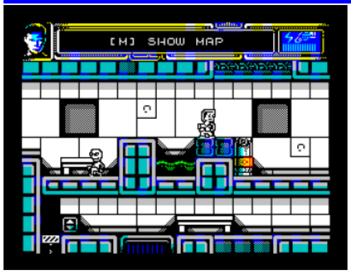
I found the levels rather easy then difficult with occasional struggle here and there. It is very easy to get into the game and do some progress in no time. The graphics are nice, detailed and it's easy to see what is going on. The sound is probably the worst part of the game. No sound effects at all, but there is a tune playing all the time which may go onto the nerves very soon. The cosmic silence would be better in this case in my opinion.

Overall, grab this game if you still love jump-and-run Games!

Download at:

`https://quantumsheep.itch.io/jumpin-jupiter`

»Marsmare Alienation« ©2020 Von Drunk Fly



Aliens are at it again. They captured you right out of your bed in the middle of the night. Because you've been sleeping, it was an easy task for them. But once you woke up, you will make them sure that they picked up a wrong guy this time.

After a short introduction we're in the action. Unarmed, inside of an alien space station, our task is to escape back home. There is a gun behind the energy field, but the way there is not that easy. So we go another direction to see what the station looks like.

There are aliens in most of the screens. At first we have to avoid them as we cannot fight without a gun. Once we get the gun, we can shoot them. Eliminated enemies (alien, but also robots, power devices and so on) doesn't appear back if we enter the screen again. There is also an auto mapping function. Once we see the map, we also see how much from our goal we already achieved and how much more is waiting for us.

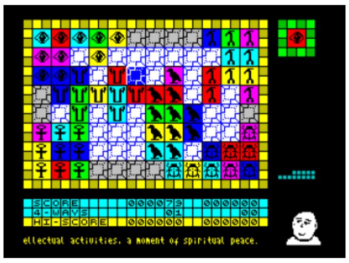
The station is fully working and that is to our advantage. We can use lifts and most important, there are computer terminals which we can use to save our position. Once we die, we are back at the last saved point. Destroying power devices is a must otherwise we wouldn't be able to pass the energy fields.

Another important things are also energy capsules for the gun, and pills that will restore our health. Once we get a space suit, we can leave the station and explore the surroundings.

The game itself is just wonderful. Animations are very smooth, graphics detailed and very nice. Controls are very accurate and the overal playability is just great. There is good music playing in the background and overal Marsmare Alienation is another great game, a must play!

`https://spectrumcomputing.co.uk/entry/36516/ZX-Spectrum/Marsmare_Alienation`

»The Legacy of the White Crane - Ishido 2« © Fanzix



Ishido 2 is a remake of an old logical game that came to the ZX-Spectrum in 2015.

After another 5 years we now get a new version that is improved in a few ways. If you like logical games, your attention should go this way.

Once you load the game, a monk will greet you. He live inside of this game and he will be your companion during all time of playing. And it is a good thing as he will not only tell you the history at the very beginning, explain the rules later but will also comment your moves during the game and let you undo them or even recommend next move. There is also a variation of the game where the monk will be playing with you. I think that more games should have a monk inside.

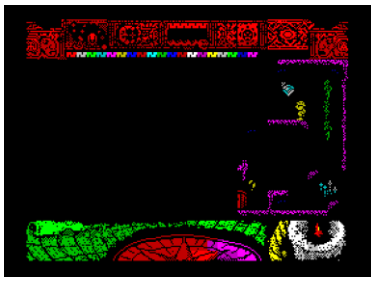
The rules are simple. We have a pile of 87 stones that have to be placed into a field of 96 cells. First 6 stones are randomly placed at the beginning of every game. Right top show us what stone is to be placed currently. The placement is always next to another stone with one simple rule: The stone must have the same colour or the same symbol of the adjacent stone(s), that's all. Sounds simple! The aim of the game is to place as much 4-ways as possible. The game end once there are no stones in the pile or no stone placement is possible. For more detailed rules, just read what monk is saying and play few games, you will get the idea soon.

The graphics are nice, detailed and we can choose from several stone symbols and different backgrounds. There is also AY music playing and sounds for beep. The controls are responsible and overall playability is good.

Download at:

https://spectrumcomputing.co.uk/entry/35329/ZX-Spectrum/The_Legacy_of_the_White_Crane

»THE LOST TREASURES OF TULUM«
© 2020 VON RETROWORKS (SEJUAN, WYZ, UTOPIAN)



Ancient treasures, myths and legends. Most People love those things, even our Hero. He is trapped inside the ancient Temple of Tulum. There should be an escape through the dark, the treasures will help finding the way out. Will you help our Hero?

Another platform game with a twist. Being in an old temple without a light is not a good thing. Do you remember **SWITCHBLADE**? A game which only show the part of the screen that is surrounding you, and you have to move around to see the full place. The temple of Tulum is similar.

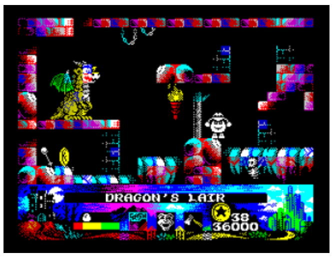
Each level contain one room. We start on a nearly black screen and have to move around the screen so we find all the treasures (it's four of them in each level). That will open the doors and we can move on to the next level and finally out of the Temple. Our hero can jump, climb, run and fight. We are not alone as there are guardians of the temple. They show up out of the dark and if we are out of the reach, they descent back into the dark. But once we uncover whole screen they have no way do disappear and we have to run to the exit. Also beware, the enemies can jump and climb too! Once a while it can happen that all light disappear. Then we have to go back to the flame that is on the place we started the level to light our candle so we can see the room again. Gold chests can be opened by attacking them sometimes that is even needed as they block the way to the treasure.

The graphics are really nice and detailed, animations are smooth and controls very responsible. The game is not the easiest ever, but together with a nice AY music it provides a good atmosphere and bring something new to the pile of modern platform games. I highly recommend to spend some time with this game!

Download at:

https://spectrumcomputing.co.uk/entry/35776/ZX-Spectrum/The_Lost_Treasures_of_Tulum

»DIZZY VIII - WONDERFULL DIZZY«
©2020 THE OLIVER TWINS



I even don't know how to start. ***DIZZY*** is back. And not in a remake or a fan game. ***DIZZY*** is back in his new, collosal adventure created by his original authors, the ***OLIVER TWINS***! And this time they worked together with many other people around the Spectrum game scene to bring us a deluxe game that will last. So, let's help ***DIZZY*** in his new adventures.

During a huge storm, ***DIZZY*** and ***POGIE*** made it into their hut at the very last moment. The storm was so powerful, they felt like the whole hut was flying away. And once the storm was over, they rushed to go out to see what damage it caused. But here comes the surprise - the hut was really blown away with the storm, and they are now in the land of ***MAGICIAN OZ***. That would be still quite OK, but when the hut landed, it landed right on the "wicked witch of the East" and her boots are the only thing that lasted after her. Once ***DIZZY*** tried those, witch's sister showed up and took ***POGIE***. Poor ***DIZZY*** now have to go across the ***LANDS OF OZ*** to find her and rescue ***POGIE.***

Please note that the game is SPECTRUM 128K only. But it take advantage of the machine and is huge. Also there is a lot of texts to read. A map can be very handy as some locations contain shortcuts or can be entered from more sides. Also, objects to collect start to pile up after a while of playing, so it is obvious the game takes quite a time to finish. Be sure to define your controls carefully as you cannot change them once you start the game.

There are new things and it is quite a lot of them, so let's take a look at those. At first it's graphics. After a black & white intro and beginning, you will be stunned by colours. And ***DIZZY*** himself is stunned too, as he is joking at the beginning of the game, really funny. The graphics is different from the old ***DIZZY*** series, some of it reminds me for the ***DIZZY VII*** remake from few years ago. ***DIZZY*** now have a back-pack visible on his back, but can still carry just 3 objects at most. He is very nicely animated, including actions like entering the doors, climbing the ladders and so on. He even sit down when you press down. Because we're not in ***DIZZY'S*** hometown ***YOLKFOLK***, he meets new creatures. And as we're now in the ***LAND OF OZ***, I am sure you will recognize at least some of them. There is a bad witch, good witch, there is the ***MAGICIAN OF OZ*** and a lot of other characters Dizzy have to deal with.

As you move around, surroundings change. The village is nice and bright, the forest is dark and creepy. This adds a lot to the overall atmosphere, the bad witch is surrounded with darkness, and the good witch lives in a nice and fancy castle.

Everything feel alive, there are creatures we should avoid (snakes and bats for example), there are characters to talk to. Water and fire and any living creatures are animated very nicely.

What about the puzzles? Well, they are logical. If you find a cog-wheel and after that walk to the screen where the bridge is missing but you see a lever, it is obvious to try it and yes, it works. And because it is a ***Dizzy*** game, the way to rescue ***Pogie*** is very long, because there are many characters that need help, and their needs varies.

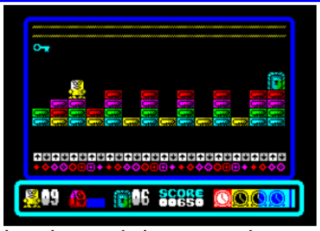
Dizzy VIII keep the tradition of action adventures, so you'll also have to jump around the platforms, avoid enemies and plan your moves. Because the controls are very accurate this is not a big deal and the gameplay is very smooth. As I already mentioned, the biggest deal here will be the orientation, as the game map grow bigger and bigger as you play.

If you are very tight with time for games but you still want to play something interesting and exceptional, ***Dizzy VIII*** is the game to choose. You won"t be disappointed and will have a great fun while playing!

Download at:

`https://spectrumcomputing.co.uk/entry/36568/ZX-Spectrum/Wonderful_Dizzy`

»DUNGEONS OF GOMILANDIA«
©2020 VON RETROWORKS



Maybe you remember ***Gommy***, a hero from the past. In ***Gommy, Medieval Defender*** he was fighting against enemy hordes that tried to invade a castle. Now he is back and he needs our help again. ***Gommy*** was thrown into a dungeon that no one ever came back from. It is not full of monsters or magic, but his current enemy is time. It is ticking really fast!

The game is a fast logic puzzle. The dungeon contain 55 rooms that we have to go through to reach the end. In each room we have to get a key that opens the door and then leave the room. It sounds quite easy, but as always there are some obstacles. First one is the movement. We can go up and down one brick. We can fall 2 bricks but cannot go up. If we fall more than 2 bricks then we lose a life. We can get 1 brick that we stand on and that will lower our height, we can also put it on the place we stand and that will raise our height. In later levels we will get also some different bricks, some will slowly disappear under our feet's, some are marked with arrows. Those can be used to move around and are very handy. But you have to be cautious, because once you stop them without possibility to step out of them, you'll definitely lose a life. There are also some »jump« stones that allow you to reach greater heights.

The graphics are nice and colourful. Movement is in attributes so be careful as ***Gommy*** can make an additional step more than desired, and that will often lead to his death. Sounds are for Spectrum 48k Beeper, and they are well made. At the beginning of each level we get a password so we don't have to play the whole game at once. That make it a great game for a couple of short runs when you are in mood to play something fast with a need of using your brain. The game also contains a relaxed mode where we have no time limit. But for this we will not get any points after finishing the levels. I recommend this game, it won't disappoint you.

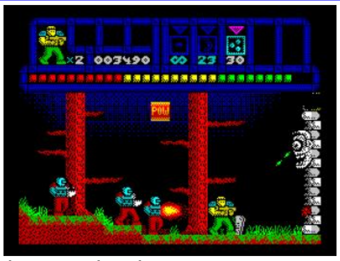
Gommy, Medieval Defender:

`https://spectrumcomputing.co.uk/entry/23487/ZX-Spectrum/Gommy_Defensor_Medieval`

Gomilandia:

`https://spectrumcomputing.co.uk/entry/35677/ZX-Spectrum/Dungeons_of_Gomilandia`

»HELL YEAH!<<
© 2020 VON ANDY PRECIOUS



The name is strange, but considering it is Andy's debut on the Spectrum, it make sense. I called it a similar name once I've played it for the first time. So, let's take a better look at it.

As usually, it is up on you to save the world. For this you have a gun and have to run over the enemy territory. As you go and eliminate hordes of soldiers, demons and monsters, you are also picking up energy boxes, first aid packs and bonus weapons. And you keep firing. Did I mentioned that you should always run? Because the game is fast! The graphics are big, colourful and very fast. There is minimum colour clashes and it all look just great. Because of this, everything move by 8 pixels and that mean speed. This is not any problem on usual terrain, but once you need to stop and need some seconds to plan your jumps, it makes things a little bit more difficult. There is no need for pixel (attribute) perfect jumps, but because enemies are constantly entering the screen, action is fast and it's not easy to focus on other things besides shooting.

I have to say that the game is actually frustrating. It took me 2 days to get into it. It need a lot of practice to be able to jump to the platforms, trees and other obstacles. Luckily, we get more powerful guns as we go. They have limited ammo, but are a good help, especially with bigger enemies.

Sounds are for Spectrum 48K, but they are quite plenty. Overall atmosphere of the game is good and as a keen action shooter I have to recommend it; give it a go if you are tired of puzzle games already. Oh, and don't use tape, the game load levels when needed.

Download at:

`https://spectrumcomputing.co.uk/entry/36497/ZX-Spectrum/HELL_YEAH` sinclair ZX Spectrum

2022 GOTO 40

1982 - 2022

→ When does the event take place?

Saturday, 30th April 2022

→ And where?

Bescot (BanK's) Stadium , Walsall, near Birmingham

→ How to get there?

By car via M6 motorway
or use public transports
(Birmingham has airport and train station)

→ Opening hours?

The Show starts at 9 am
and ends at 9:30 pm (Party)

→ What about the costs?

Day-Ticket adult: 20 GBP
Day-Ticket 2 adults: 35 GBP
Day-Ticket children 5 - 16 years: 15 GBP
Day-Ticket family: 50 GBP
Party-Ticket (only for evening): 10 GBP

→ What will await me there?

Guests from 40 years Spectrum history:
Tim Gilberts, The Oliver Twins, Sandy White...
Opening song by Mark Hibbet,
ZX Fair sales stalls, Spectrum Next raffle
and much more

→ Where to buy tickets and get additional information?

https://spectrumevents.website/